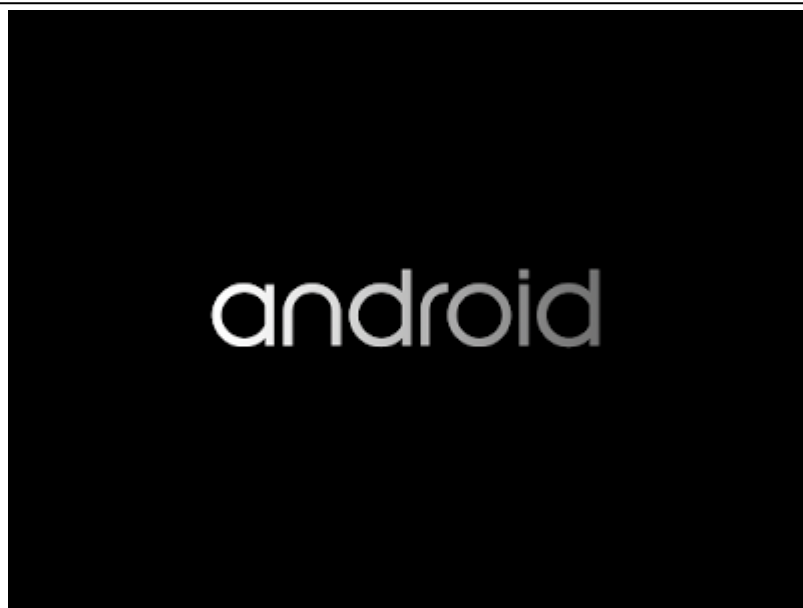
[ATUALIZADO] Análise: Android x86 Oreo (Parte II)

Tela de inicialização do Android x86, utilizada em compilações oficiais desde o Lollipop.

Introdução

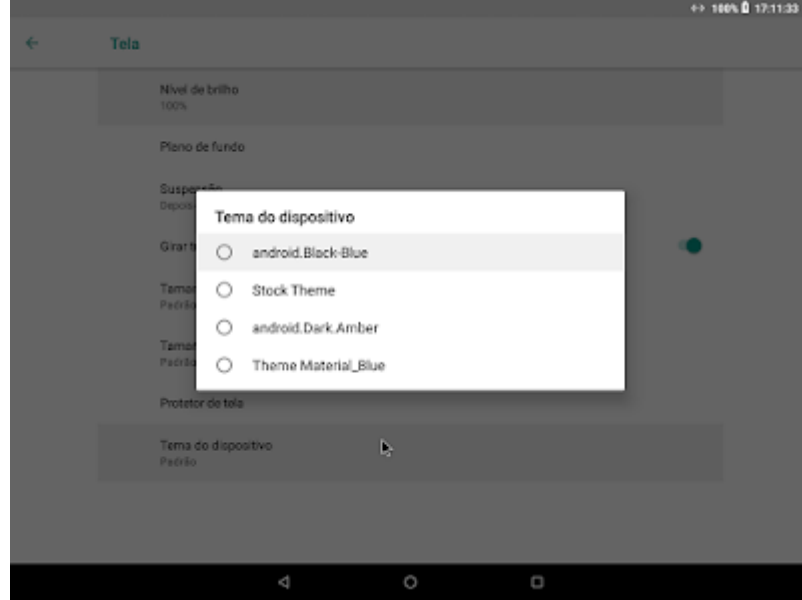
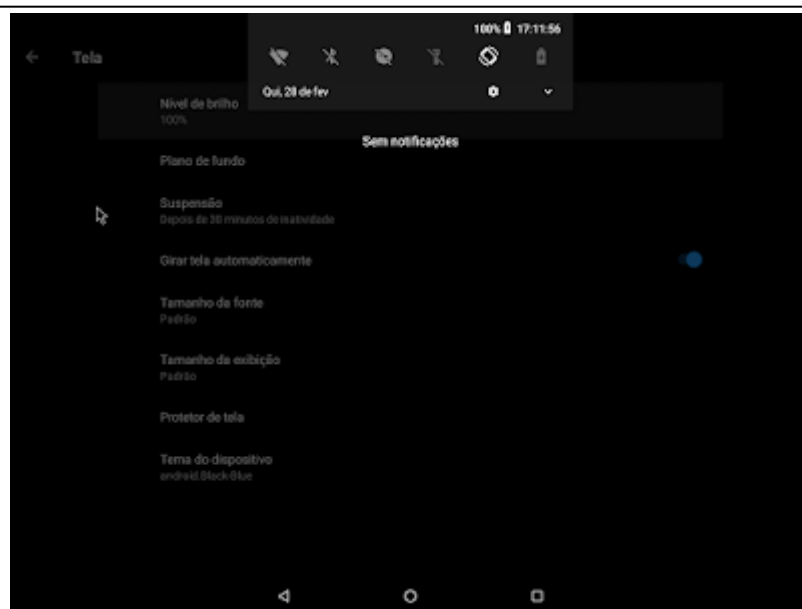
Para facilitar a leitura, a análise sobre o Android x86 Oreo foi dividida em duas partes. Caso tenha pulado de paraquedas por aqui e quiser ver outros detalhes das compilações desta edição do SO da Google, incluindo como baixá-lo, a Parte 1 pode ser acessada clicando **aqui**.

Esta segunda parte abordará os seguintes assuntos:

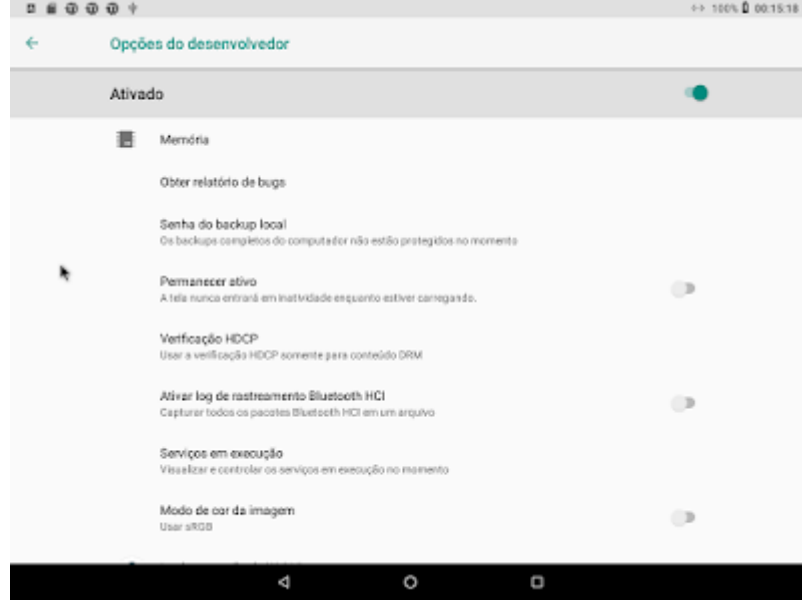
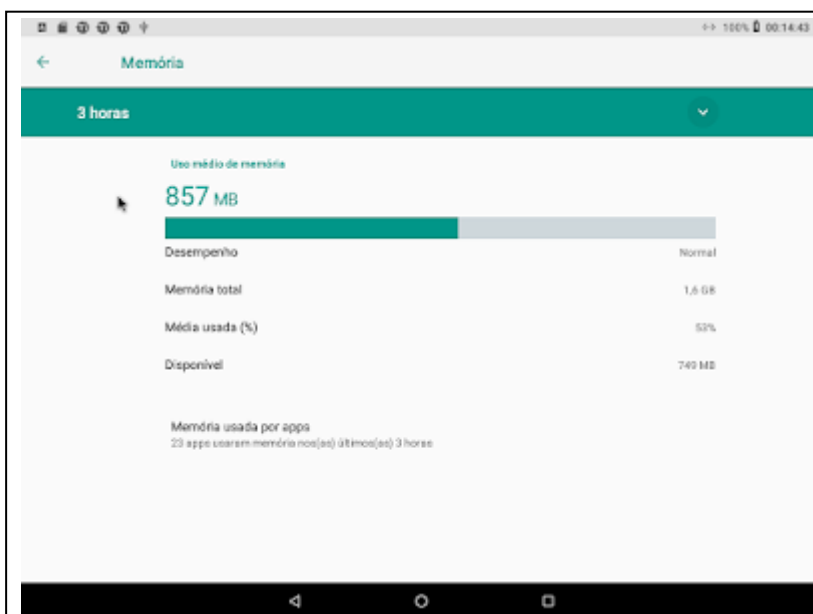
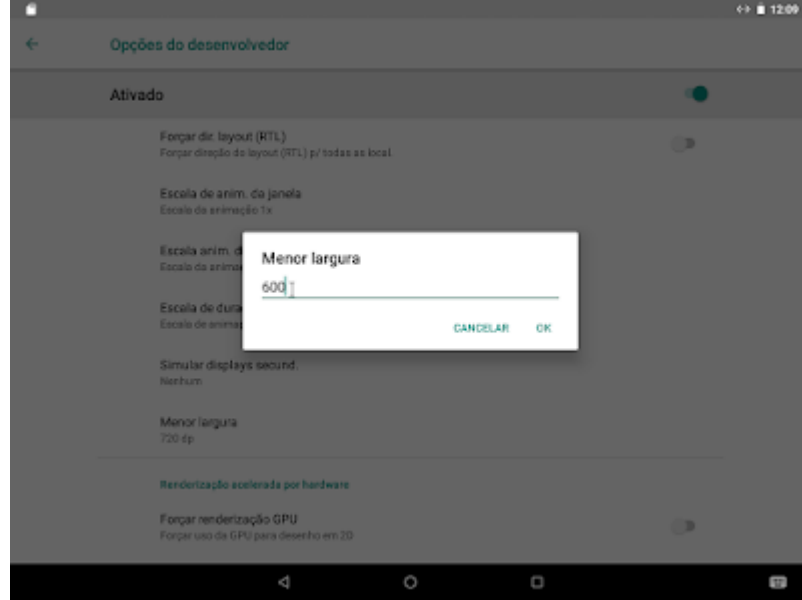
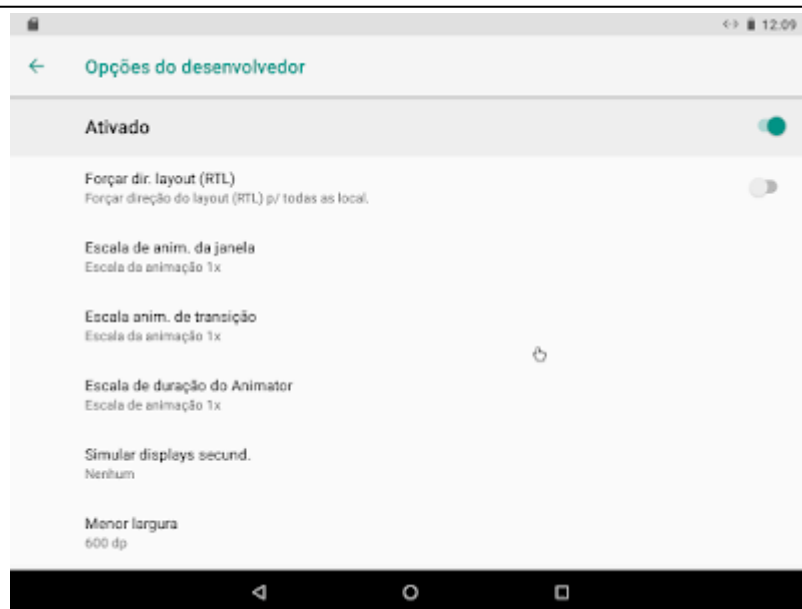
Instalação de Temas

Faz muitos anos que a inclusão do tema escuro no Android Puro vem sendo pedida pelos usuários, mas até agora a Google não atendeu esta solicitação tão requisitada (o que é uma regressão, na verdade, o qual detalharei nas Conclusões).

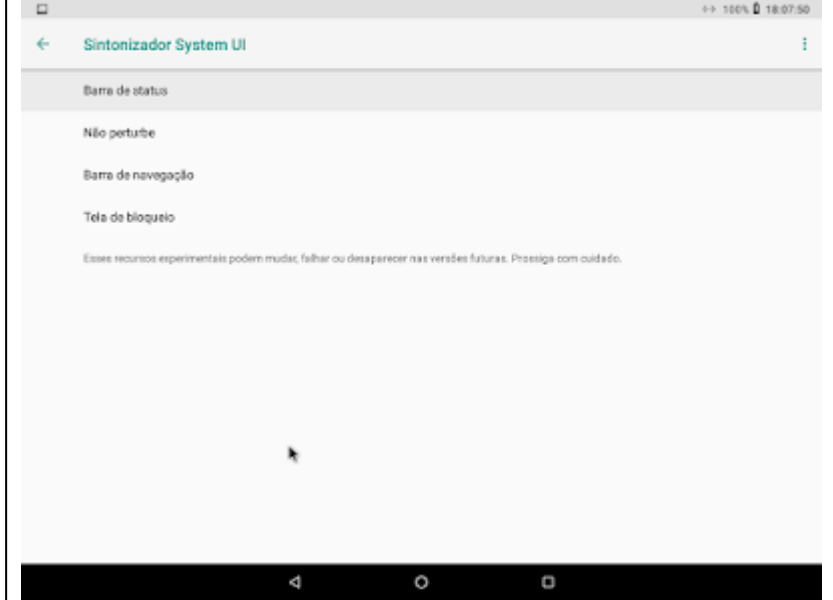
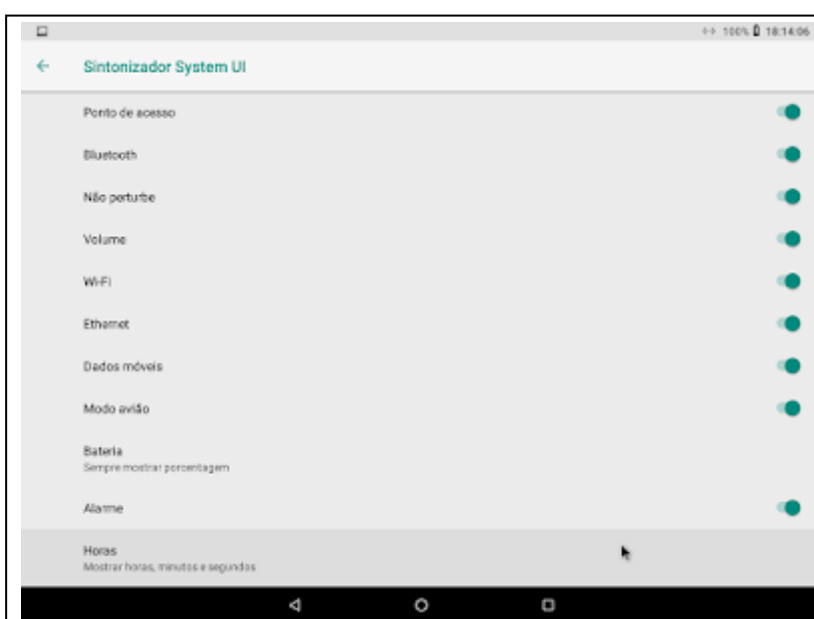
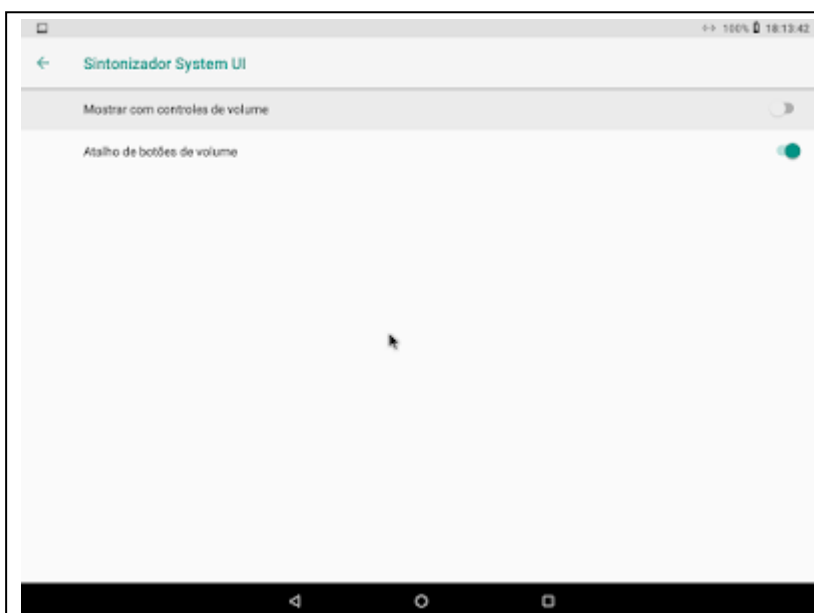
Contudo, o Android Oreo inclui um motor que permite a instalação de temas, permitindo a aplicação de variações escuras sem o uso de *mods* que exigem acesso *root (vale ressaltar que o recurso foi removido no Pie).*

Pelo recurso estar restrito aos desenvolvedores e fabricantes, acaba-se dependendo deles para a utilização do recurso (desabilitado por padrão), tendo que obter os temas por meio de fontes não oficiais (os arquivos necessários podem ser obtidos clicando aqui).

Após instalar os temas (no mínimo, dois - é recomendável instalar o *Stock Theme*, caso queira reverter, já que a opção não aparece por padrão), é preciso selecionar a última opção que vai aparecer em Tela e selecionar o desejado.

É um recurso muito útil, principalmente em telas AMOLED, sem falar das variações de cores que podem distinguir seu dispositivo dos outros, contudo, os temas não são 100% perfeitos, podendo conter elementos que não estejam muito bem adaptados ao ambiente.

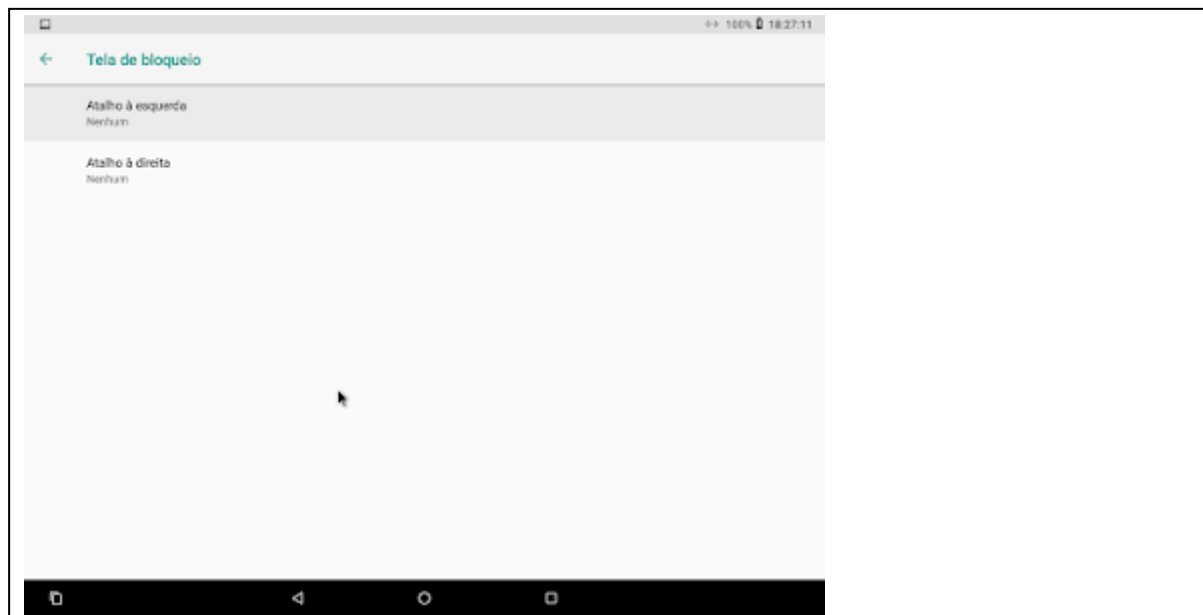
Opções de Desenvolvedor

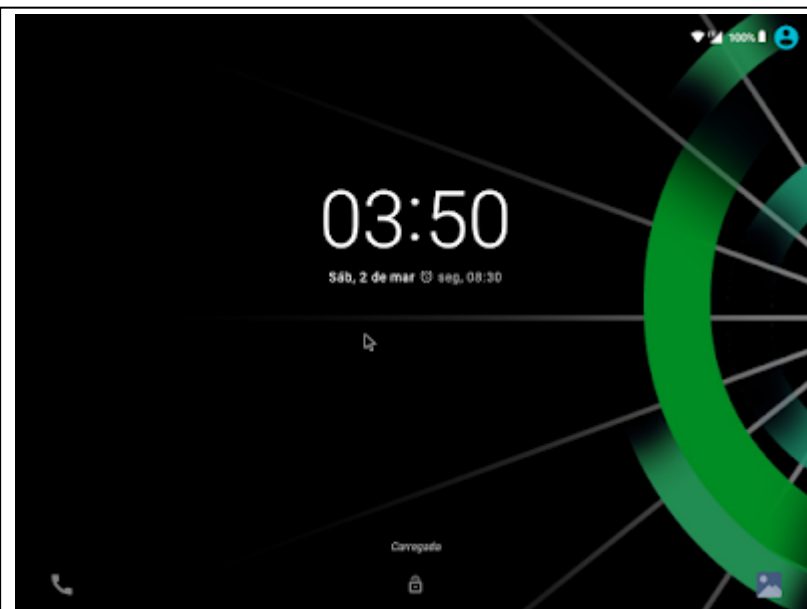
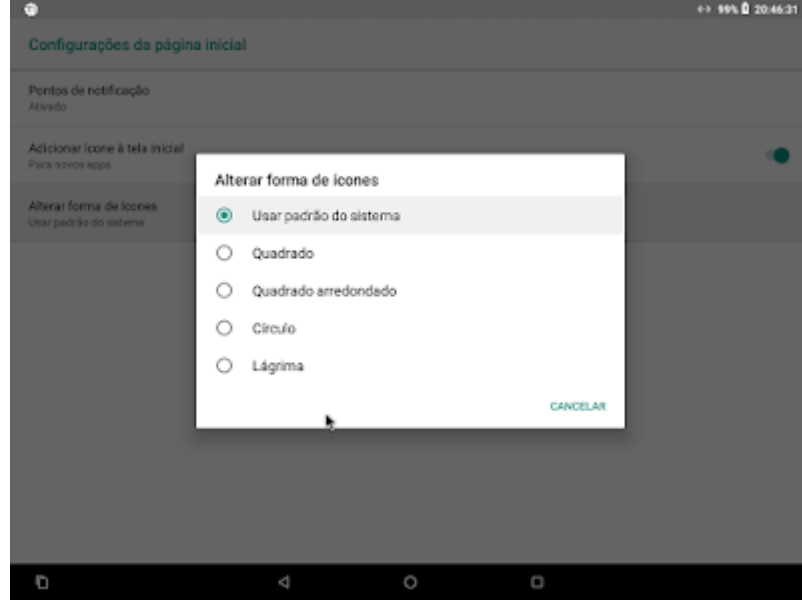
Como dito antes, segue basicamente o mesmo funcionamento, com poucas mudanças desde a versão anterior.

Logo de cara, algo bastante intrigante: se desde o Marshmallow era possível acessar o Serviços em Execução (o mesmo do Lollipop, herança das versões anteriores do Android), o Oreo, por incrível que pareça, disponibilizou uma segunda opção: Memória.

Desmembrado das Configurações e, desde então, escondido do usuário comum, o recurso é o mesmo que tinha sido remodelado no Marshmallow.

Uma outra opção interessante é o ajuste de resolução do sistema, que permite adaptar os elementos das telas, aumentando ou diminuindo o tamanho, semelhante como ocorre no Windows - ou seja, não altera a resolução em si (adiantando que este recurso foi movido para a opção Telas no Android Pie - versão 9; SDK 28).

Perceba como os elementos do sistema foram ampliados e empurrados para mais próximo da borda da tela, alterando o valor da menor largura para abaixo do padrão (no caso, 600 - o padrão é 720).

Sintonizador System UI

O recurso escondido, que personaliza determinados elementos do sistema, continua no Oreo, recebendo novas opções.

Sintonizador System UI no Android x86 8.1 Oreo R1.

Na Barra de Status, desde o Nougat, permite a personalização de quando exibir a porcentagem de bateria, além do formato da hora, que agora exibe os segundos (sem o uso de *mods* de terceiros que exigem *root*, como o Gravitybox).

O Não Perturbe é um modo de som que permite sua configuração pelos

botões de volume, não muito útil em PCs (talvez as teclas do teclado dedicadas não suportem o recurso).

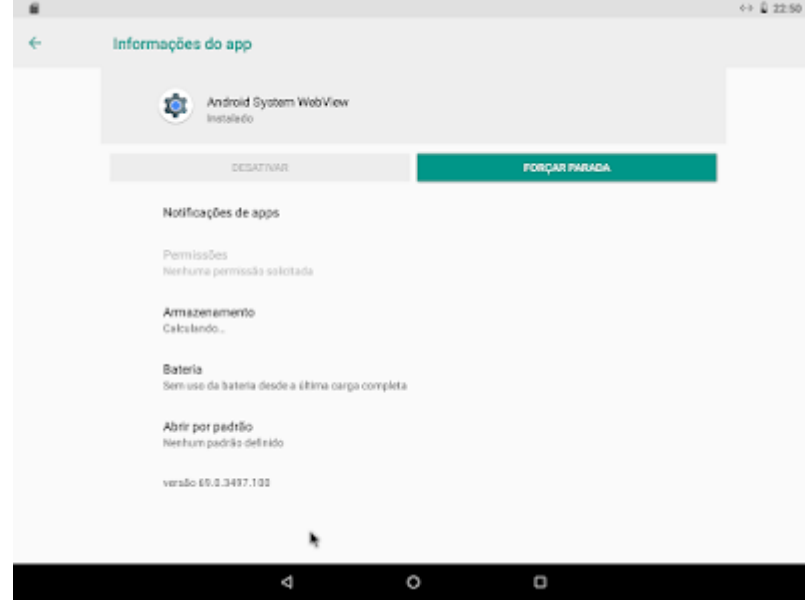
Na opção Barra de Navegação, por alguma razão, ao abrir a tela, imediatamente as opções Código de tecla à esquerda, Ícone à esquerda; Código de tecla à direita; Ícone à direita somem (só lentamente que foi possível capturar a imagem acima), sem saber qual a sua função, podendo adicionar comandos adicionais, como pode ser visto, no canto da barra, o ícone da área de transferência.

E a Tela de bloqueio também permite a adição de ícones (incorporando um recurso da Samsung existente desde a TouchWiz 6.0 no Android Marshmallow, mas que tem base desde o Jelly Bean).

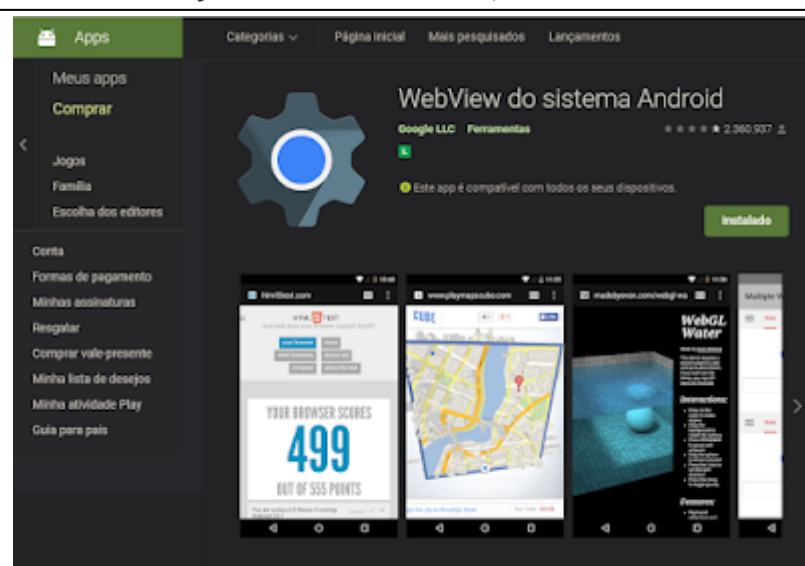
Ao selecionar os atalhos desejados, surgem as opções que permitem que, ao movimentar o atalho, automaticamente a tela é desbloqueada (repare que o título é trocado, exibindo como "Atalho à direita" erroneamente).

Como pode ser visto acima, os botões selecionados na tela anterior são exibidos aqui (vale ressaltar que não é nada prático fazer o movimento de desbloqueio usando o mouse – isto porque não há suporte para uso com teclado).

Além disso, ao habilitar o recurso, automaticamente, ao acessar a configuração da *launcher*, haverá uma nova opção, que é uma das características do Oreo, que é o formato dos atalhos.

Este é um exemplo da gaveta de *apps* com o formato dos ícones circular.

Apps Embarcados

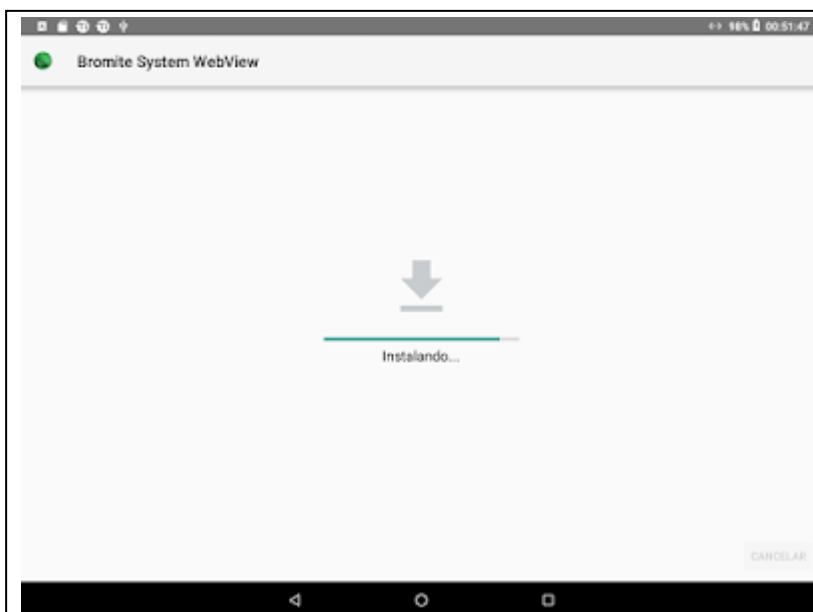
WebView

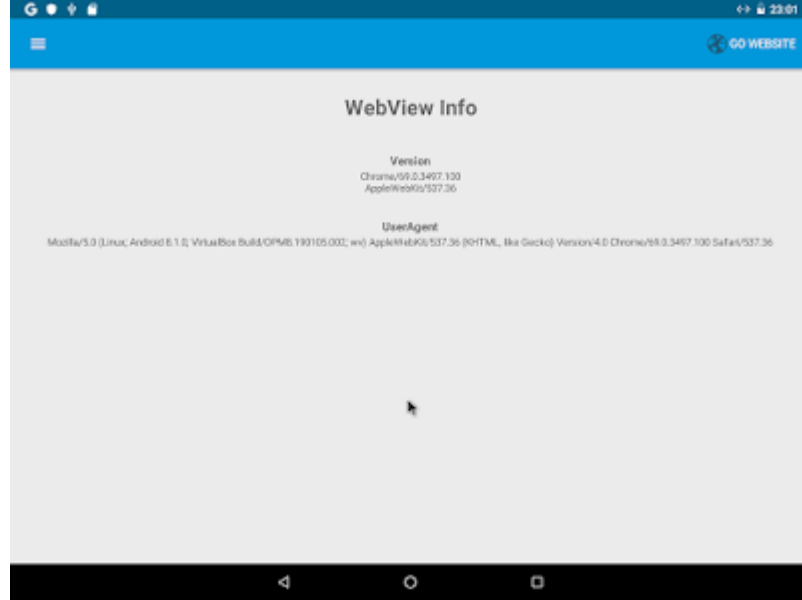
A partir do Oreo, o projeto deixou de incluir a versão alternativa do recurso, atrelado à compilação e que, portanto, não podia ser atualizado pela Google Play Store a menos que se instalasse uma atualização do sistema, ainda que não trouxesse a versão mais recente do recurso, mas sim a relacionada com a versão do Android.

Por padrão, as compilações do Oreo trazem a versão 69 do *app* disponibilizado pela Google.

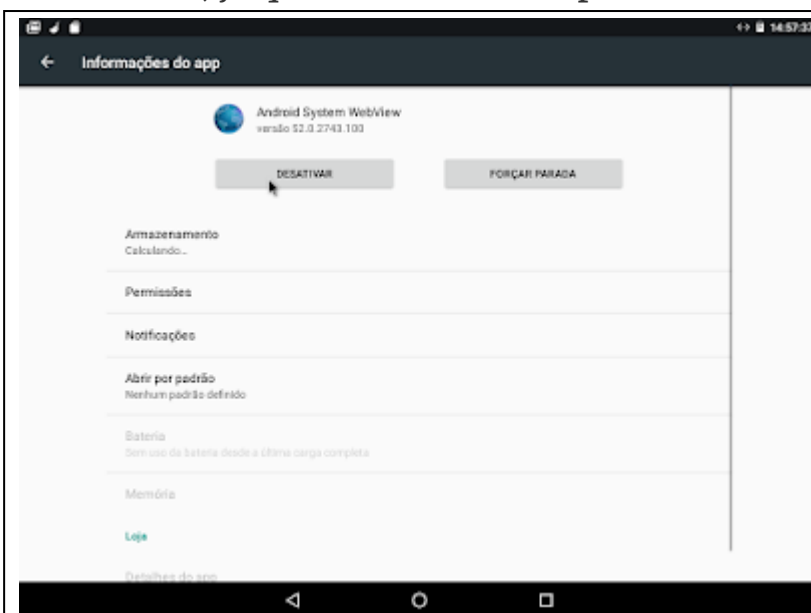
Contudo, em testes internos, constatei uma brecha relativamente grave relacionada ao Android, que não é muito difundida por aí. Antes, vale um breve contexto histórico para entender como se deu a evolução deste recurso ao longo dos últimos anos, desde a descoberta de uma falha que ficou famosa por deixar milhões de dispositivos Android, com a versão Jelly Bean (4.1.x a 4.3; SDKs 16 a 18) e anteriores, sem correção.

Desde o KitKat (versão 4.4.x; SDKs 19 e 20) o WebView passou a ser desenvolvido como parte do projeto Chromium e, para facilitar sua manutenção, a Google o transformou em um aplicativo para ser gerenciado a partir da Play Store (ao invés de atualizá-lo com o sistema - o que exclui, conforme citado anteriormente, a versão AOSP, utilizada pelo projeto Android x86 até então), a partir da versão Lollipop (versão 5.x.x; SDKs 21 e 22).

A partir da versão Nougat (versão 7.x.x; SDK 24) a Google fez mudanças no funcionamento do WebView, onde dá a possibilidade de utilizar o próprio Chrome como responsável pelo recurso, deixando o app Android System WebView preterido e desativado como padrão.

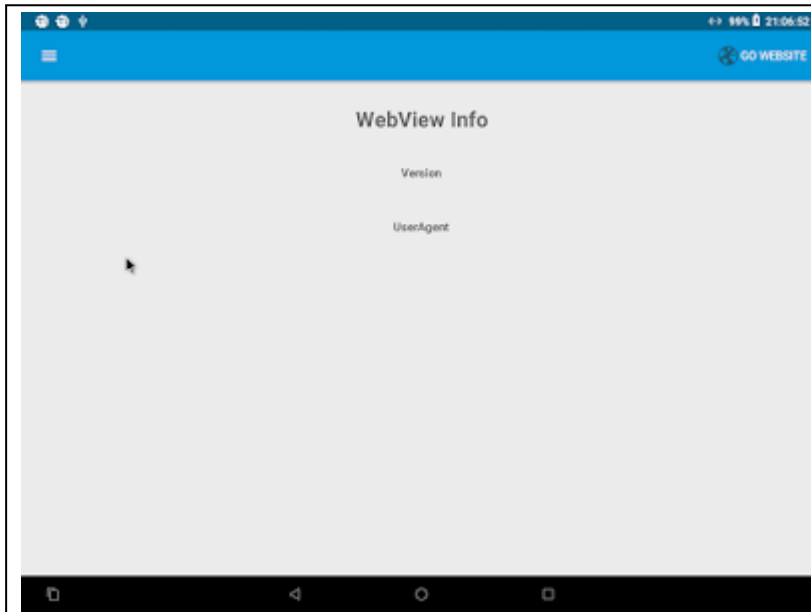
Desde o Nougat também é possível alternar o componente a ser utilizado como WebView, através da opção "Implementação do WebView", em Opções de Desenvolvedor.

Pois bem, uma versão alternativa aos apps da Google e AOSP é desenvolvida pela Bromite, o qual é baseado no Chromium, com a diferença do AOSP que é atualizado constantemente (as vezes, até mais que o da própria Google) e com *patches* que aprimoram seu funcionamento.

Como o nome do APK é o mesmo da versão AOSP (*com.android.webview*), inclusive aproveitando o ícone, ele entrava em conflito com a versão presente no Android x86 (até as compilações do Nougat - desconsiderando as não oficiais, geradas a partir do código fonte do projeto, sem alteração) e não instala, o que exige a remoção do aplicativo nativo (por sinal, até recomendado, pelas razões que detalharei a seguir).

Como no Lollipop e Marshmallow (versão 6.0.x; SDK 23) não há como gerenciar o WebView, ao instalar o aplicativo da Bromite, automaticamente se decide pela desativação da versão nativa (seja da Google ou AOSP), já que o sistema permite a operação.

Para minha surpresa, ao fazer um teste utilizando o *app* WebView Test, pode-se constatar que o WebView utilizado era o que eu tinha acabado de "desativar" e, pior, na versão de fábrica, evidentemente defasada e sujeito a falhas de segurança.

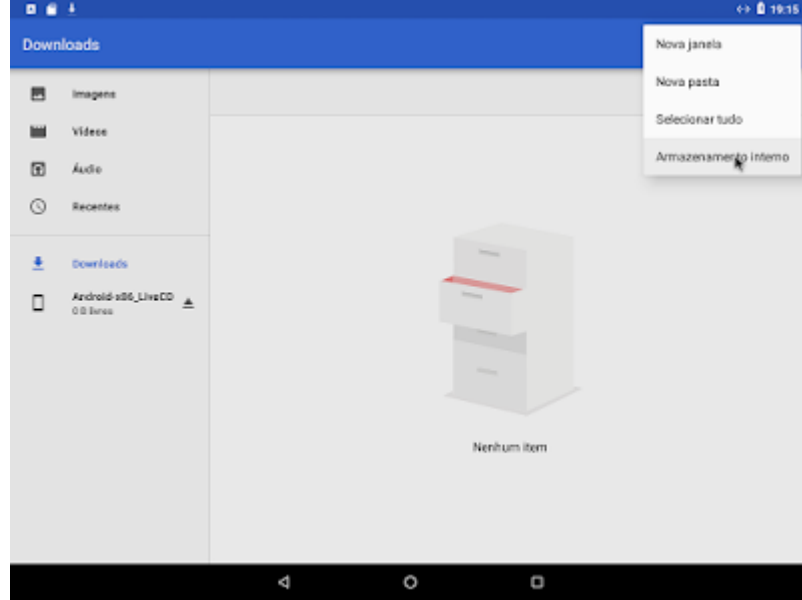
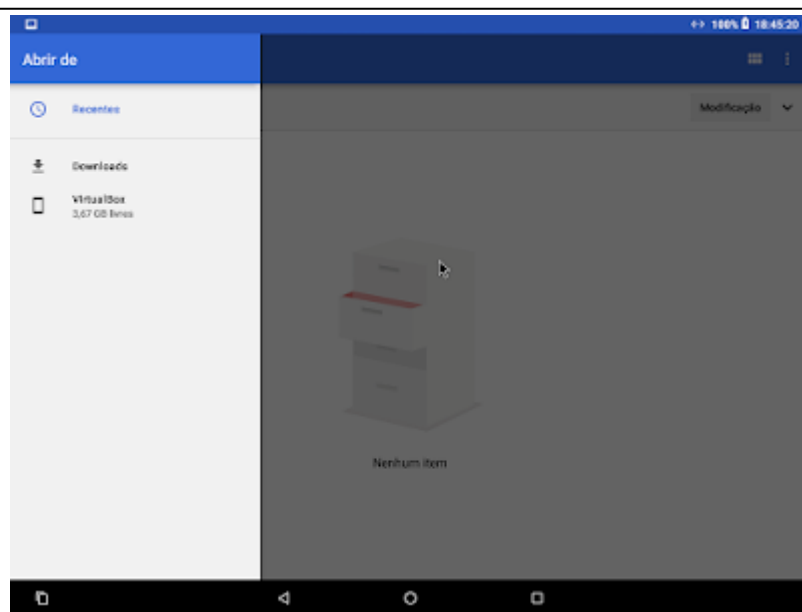
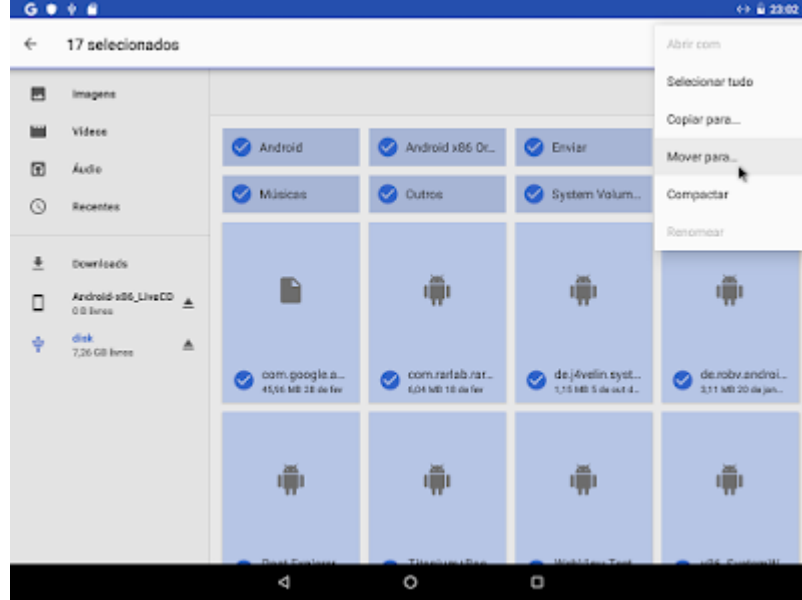
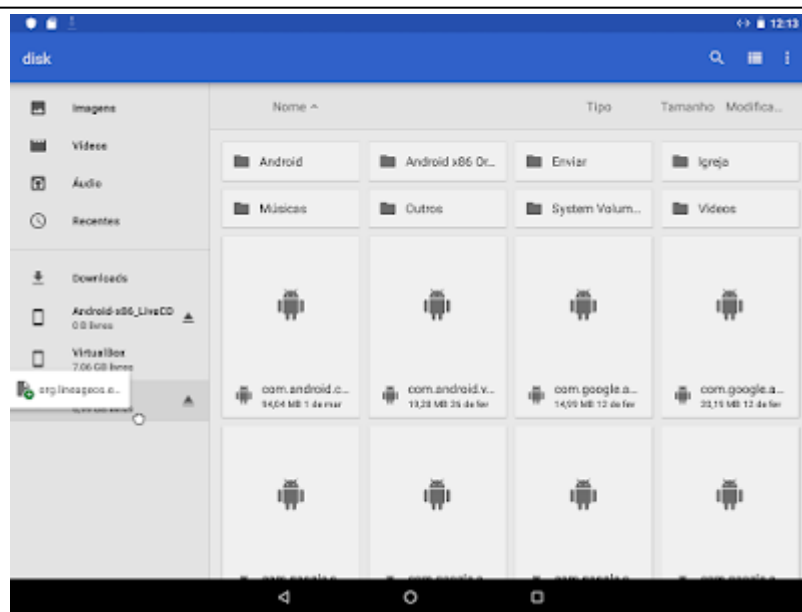
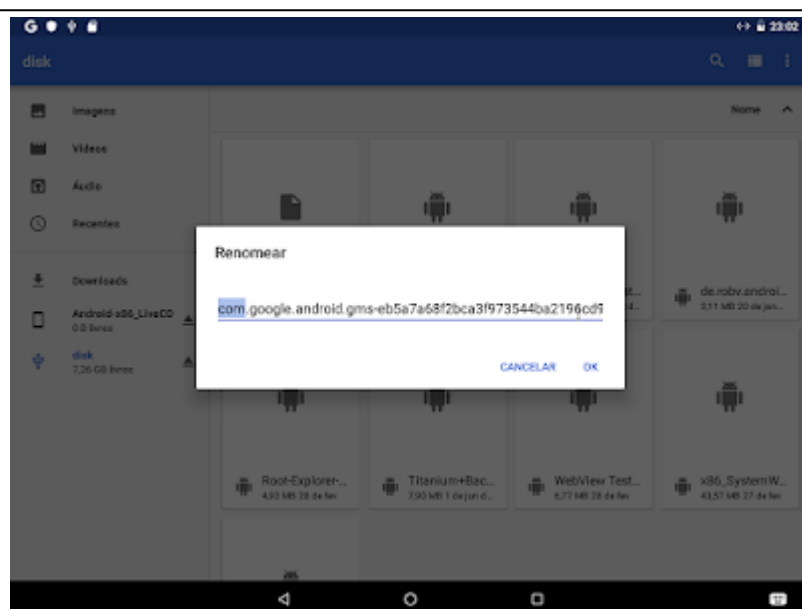
Na prática, além de o recurso não poder ser desativado, o WebView da Bromite simplesmente é inútil nas versões 5 e 6 do Android caso o aplicativo nativo não seja removido e este fique em seu lugar, como *app* do sistema (o que, obviamente, exige permissões nível *root*, o que a maioria dos usuários não tem acesso). A partir do Nougat, o mesmo se aplica se tanto o Chrome quanto o Android System WebView estão desabilitados e o Bromite, quando instalado como *app* do usuário, não selecionado, já que o sistema dará preferência às soluções nativas.

Portanto, milhares de dispositivos Android nas versões citadas podem estar vulneráveis caso desativem o Android System WebView ou mesmo negligenciando suas atualizações.

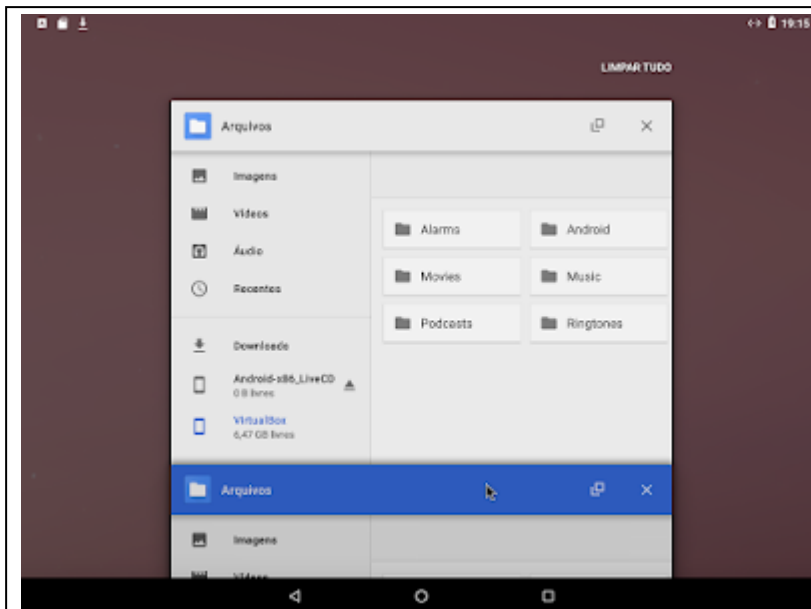
Contudo, se simplesmente desinstalá-lo, corre-se o risco que quebrar determinadas funcionalidades do sistema, como *apps* que dependem do recurso, seja o Opera Mini, Firefox Focus ou PWAs em geral. Soma-se a

isso, um problema conhecido pela Google que envolve cenários de vários usuários no mesmo dispositivo na versão Oreo (mas que pode afetar, de alguma forma, versões anteriores, considerando o relato até aqui), onde o Chrome é desabilitado por padrão, em um Perfil de Trabalho, mas o Android System WebView também está desabilitado, causando *crash* no *apps* que dependem do recurso.

Ao utilizar o WebView Test sem o *app* padrão que fornece o recurso (seja a versão da Google ou AOSP), o *app* simplesmente trava, o que pode acontecer com outros apps instalados que dependam dele.

Por esta razão, é muito importante manter o WebView (ou o Chrome na versão 7 e posterior do Android) habilitado e devidamente atualizado (conforme recomendação dos próprios desenvolvedores, que desencorajam sua suspensão) ou, caso tenha acesso *root*, remover a versão de fábrica (estando ciente dos riscos de não ter o recurso no sistema).

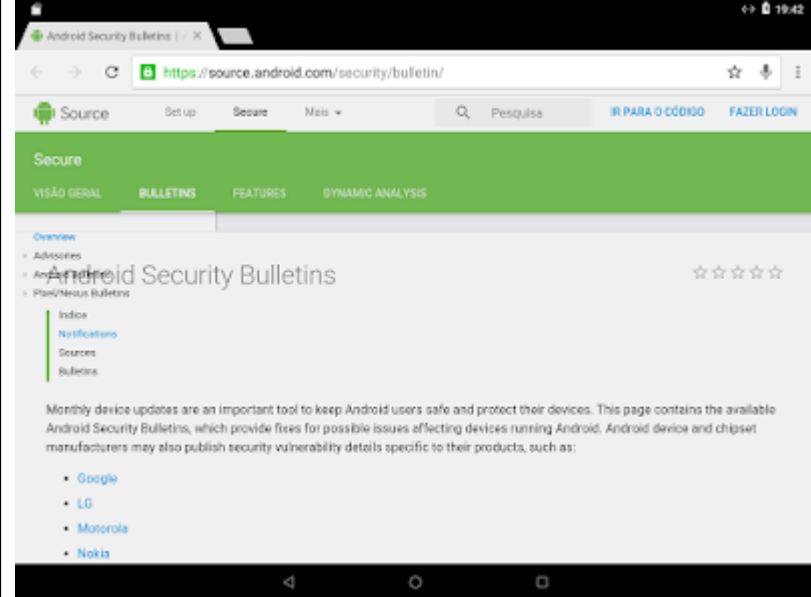
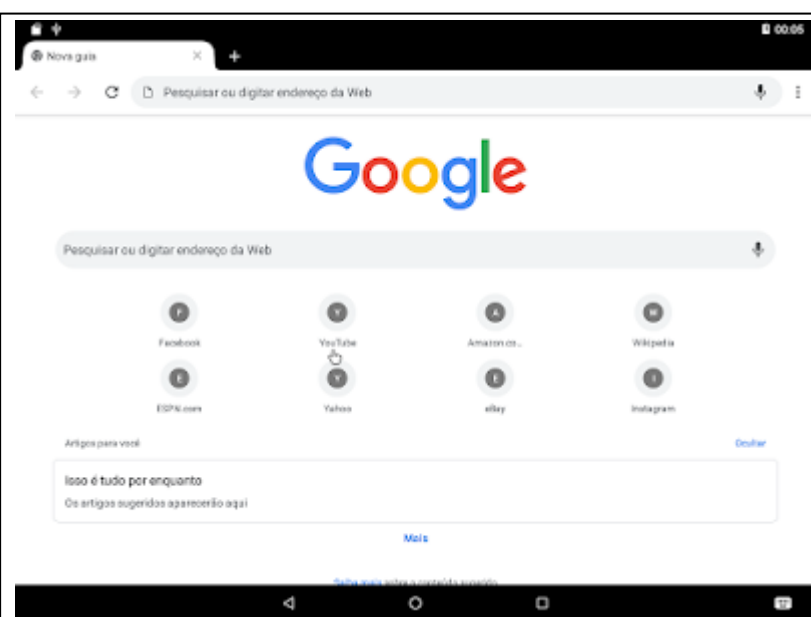
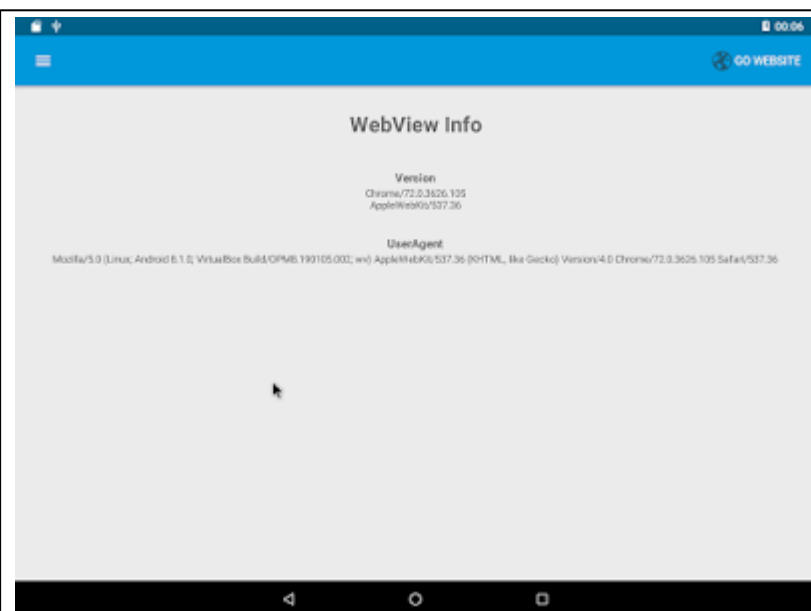
Arquivos

A partir do Oreo, as versões puras do Android, incluindo o AOSP, passaram a incluir o atalho do Gerenciador de Arquivos nativo na gaveta de aplicativos que, basicamente, substitui o atalho para Downloads (disponível desde as versões mais antigas do Android até o Nougat, que transformou o recurso em parte do gerenciador).

Visualmente, ele mudou pouco, em relação à versão anterior do Android, ganhando a cor azul na barra de título e permitindo, por exemplo, habilitar ou não a exibição do Armazenamento Interno ou ejetar um dispositivo externo; um menu hambúrguer (expandido no modo paisagem padrão do sistema) que integra atalhos para localização de determinados arquivos (o que inclui o suporte a aplicativos como o Drive).

Este é o modo de busca utilizado pelos outros aplicativos, onde o menu hambúrguer fica mais evidente.

Se no Marshmallow o *app* só permitia a cópia e exclusão dos arquivos, desde o Nougat passou a ser possível mover e renomear, tornando o recurso uma opção mais decente frente às soluções de terceiros.

O *app* também suporta o recurso arrastar e soltar de arquivos, facilitando a cópia.

Para poder renomear o arquivo, é necessário selecionar no ícone inferior

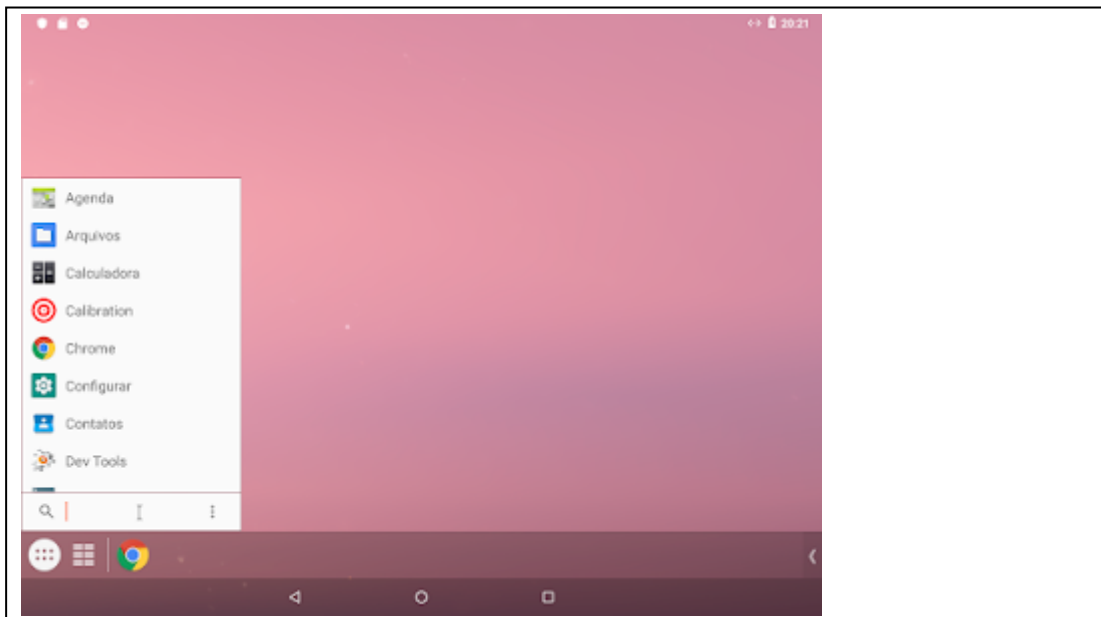
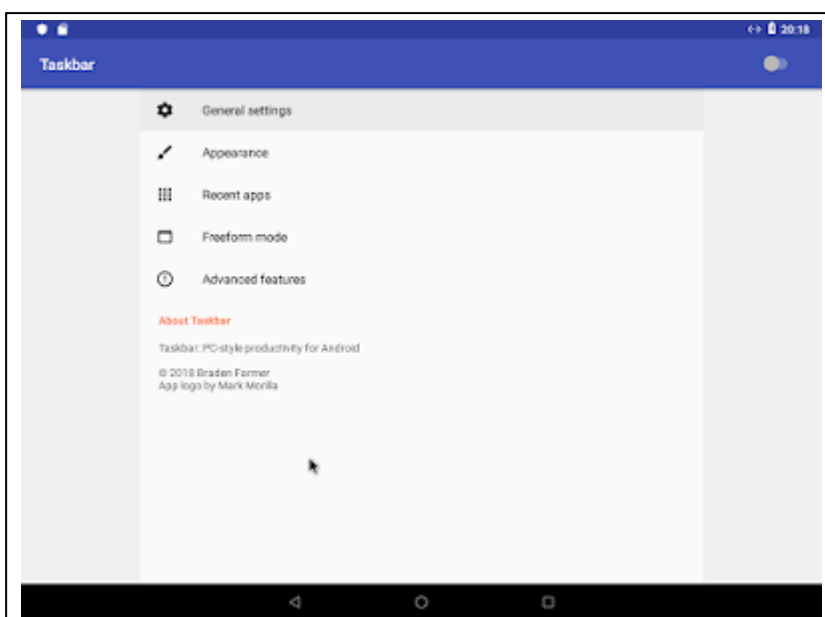
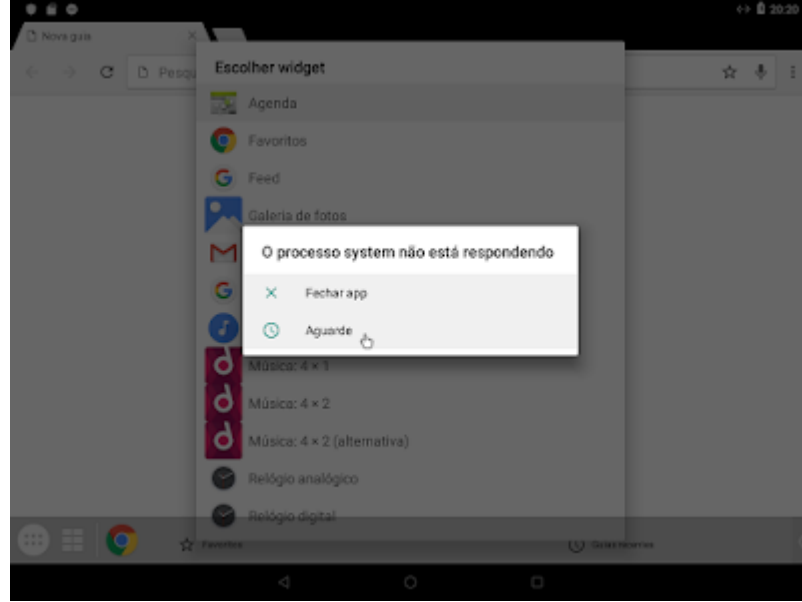
esquerdo do arquivo, para o *app* entrar no modo de seleção única, habilitando a opção.

Também desde o Nougat se tornou possível abrir mais de uma instância do gerenciador, como pode ser visto na imagem acima.

Chrome

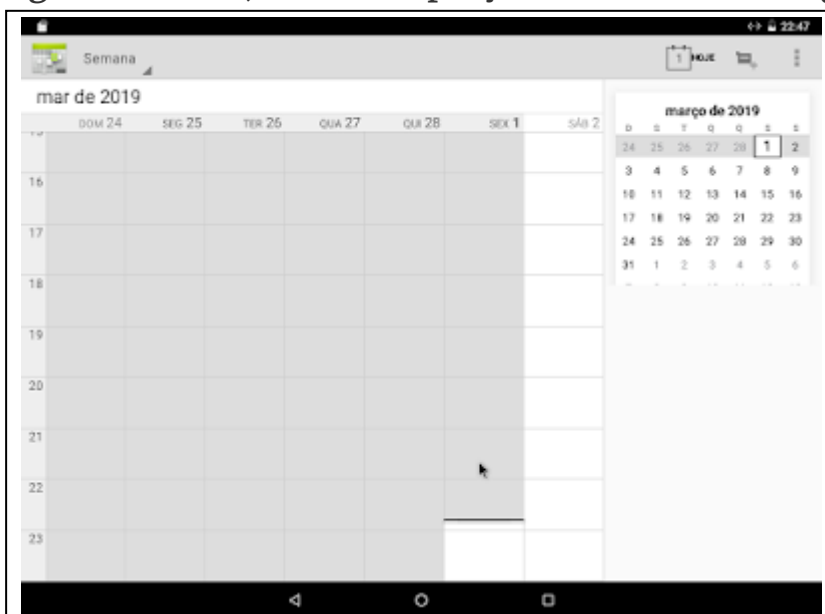
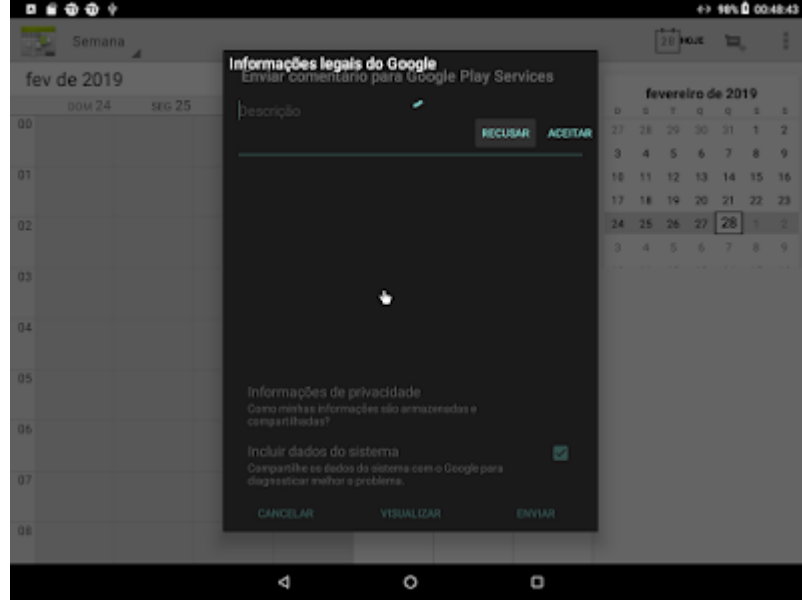
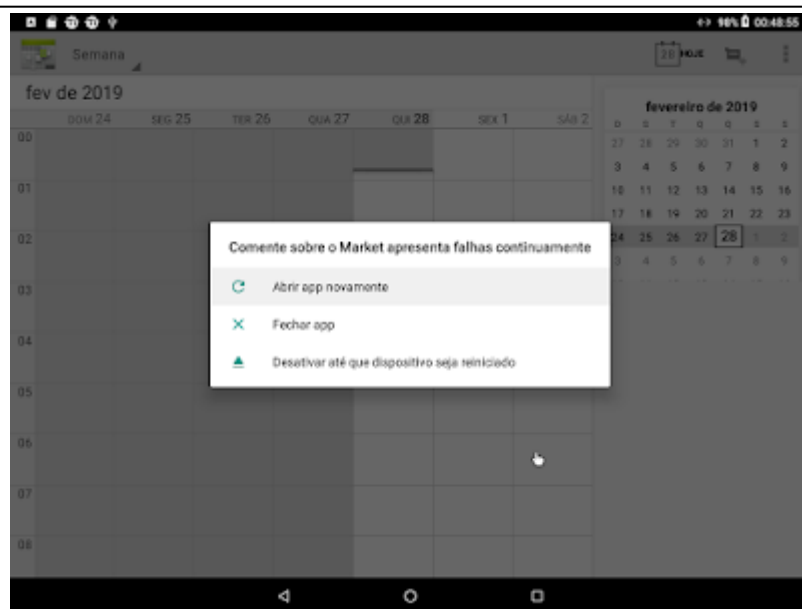
O navegador padrão da Google continua sendo embarcado nas compilações do Android x86, aqui vindo na versão 50.

Chrome no Android x86 8.1 Oreo R1 (repare que, mesmo embarcando uma versão muito antiga para o sistema - deveria vir, no mínimo, com a versão 63 - talvez por razões de compatibilidade, considerando os problemas do passado - ele consta como sendo de 2019, o que é estranho, já que o navegador não costuma alterar automaticamente a informação).

A versão nativa funcionou sem grandes problemas.

Como a versão instalada não suporta o modo WebView (conforme mostrado no tópico sobre o tema), atualizei o Chrome para a compilação 72 (a mais recente até a data de publicação do artigo), que sofreu alterações na interface, suporta o ícone personalizado, dentre outras funcionalidades; também funcionou sem sustos.

Utilizando o WebView Tester para testar o suporte do Chrome no modo citado, o *app* reconheceu corretamente (no caso, o Android System WebView, mantido na versão original para este caso, foi automaticamente desabilitado pelo sistema, conforme previsto).

Taskbar

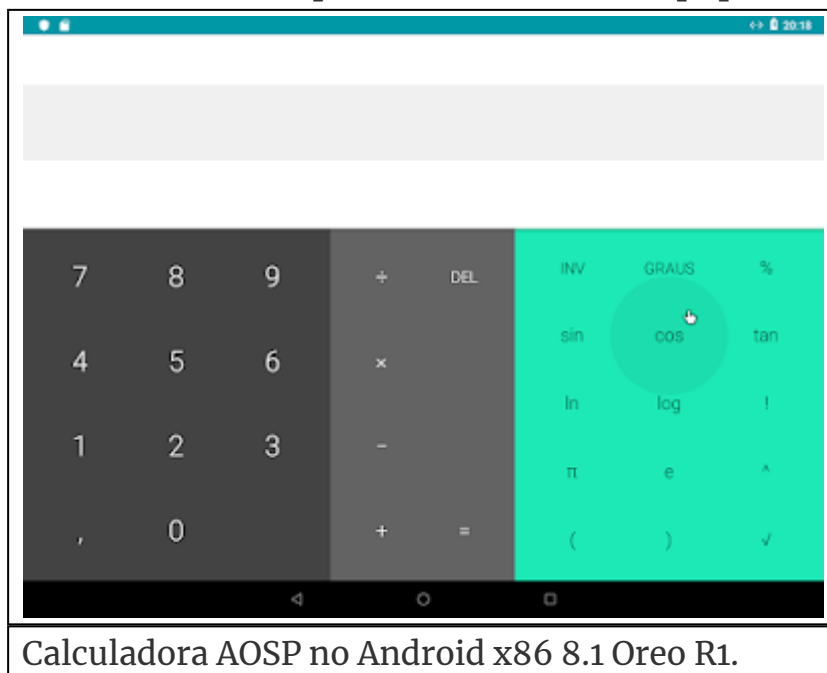
Como citado anteriormente, a partir do Nougat, o projeto passou a incluir uma versão alternativa do aplicativo criado por Braden Farmer, que simula uma área de trabalho, a fim de fazer frente aos projetos derivados do Android-x86, Phoenix OS e OPENTHOS, só que mais simples e básico (o *app* também é utilizado no Bliss OS).

Taskbar 4.0.1 (versão nativa modificada) no Android x86 8.1 Oreo R1.

Tela de configurações do Taskbar.

Entretanto, o *app* não suportou o uso de *widgets* no ambiente, causando travamentos.

Agenda

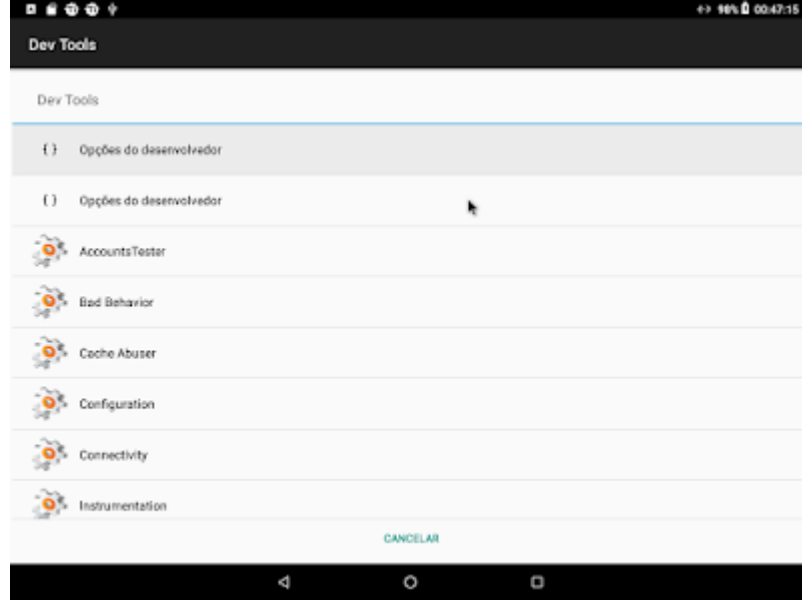
Por incrível que pareça, o projeto continua incluindo a antiga versão da Agenda do ICS, sem a adaptação do Material Design do Oreo.

Agenda AOSP no Android x86 8.1 Oreo R1.

Ao tentar forçar a autenticação e travar por causa do *bug* do Play Services, uma surpresa: é exibida uma variação diferente da tela de *feedback*, totalmente mal adaptada para o Oreo e fora dos padrões de como, normalmente, deveria ser exibida.

Além disso, o recurso (que, por incrível que pareça, é conhecido como "Comente Sobre o Market") também acaba travando, impedindo o envio do comentário (aproveitando, a tela de erros, reformulada a partir do Nougat, ganhou novas funções, como a opção de reiniciar; enviar comentário - quando suportado; fechar - antes, automático e a única opção; além de desativar o *app* até a próxima reinicialização).

Normalmente, a tela de *feedback* sugerida pelo sistema seria esta (ligada à Play Store), o que leva a seguinte pergunta: porque a Google, nesta altura do campeonato, ainda mantém, nos códigos do sistema, um recurso legado que, provavelmente, só os aplicativos AOSP mais antigos ainda utilizam?

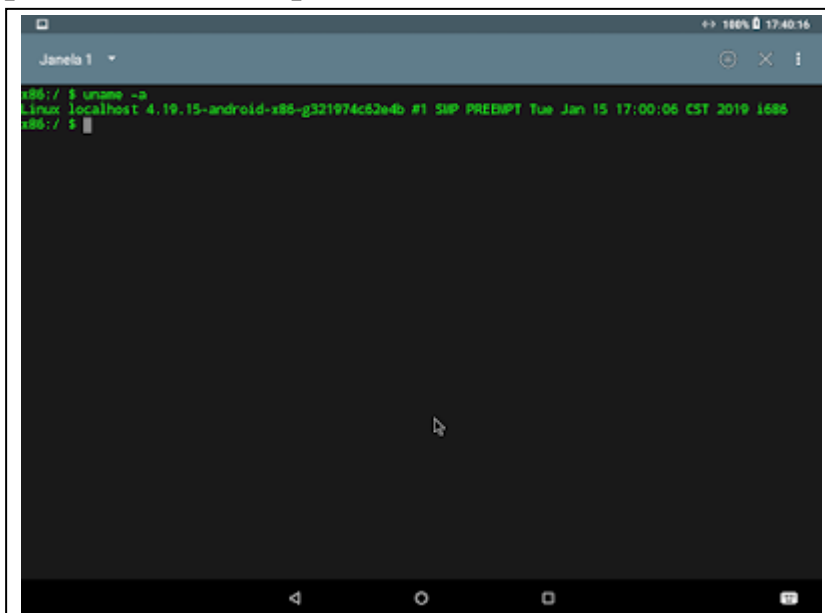
Calculadora

É a mesma versão presente desde o Lollipop.

Calculadora AOSP no Android x86 8.1 Oreo R1.

Dev Tools

Permanece o mesmo desde o Nougat, quando teve o fundo mudado para a cor branca (antes era escuro), mas ainda continua somente em inglês.

Repare que o atalho para as Opções de Desenvolvedor fica duplicado quando habilitado na Tela de Configurações, não exibindo-o quando desabilitado.

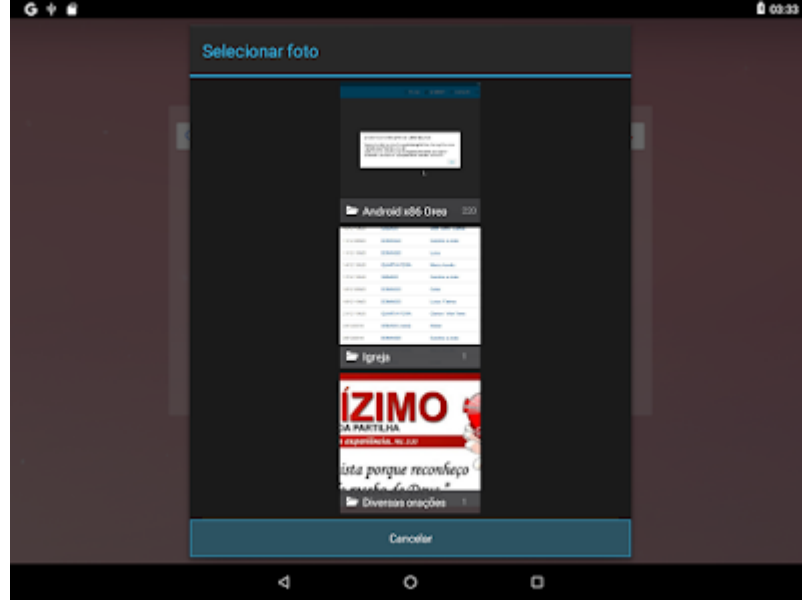
Emulador de Terminal

Não recebe atualização por parte do desenvolvedor faz algum tempo, permanecendo aqui na versão 1.0.70.

Emulador de Terminal no Android x86 8.1 Oreo R1.

Fora as duas opções de visualização de consumo de memória nas Opções de Desenvolvedor, também é possível utilizar o comando *top* para tal... :D

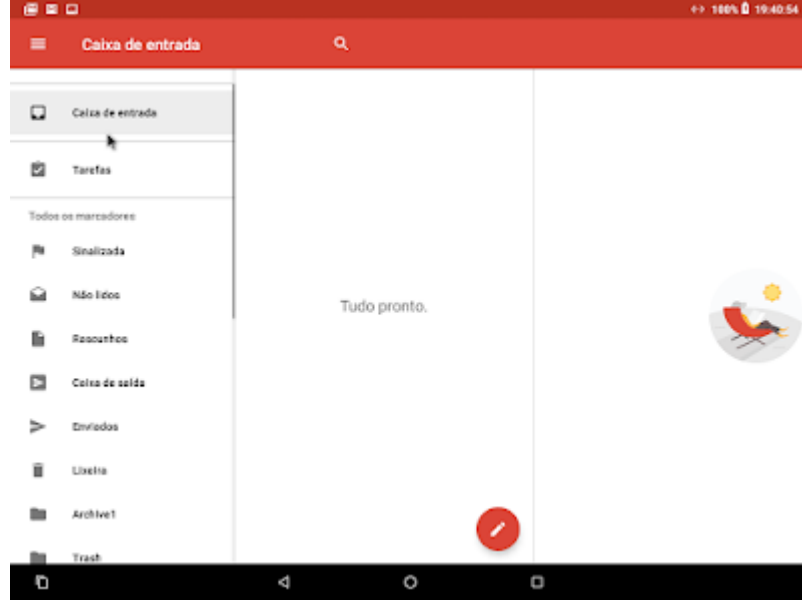
Galeria

Sem comentários... a única mudança (fora o ícone atualizado) foi a mensagem *toast*, que agora está branca por padrão.

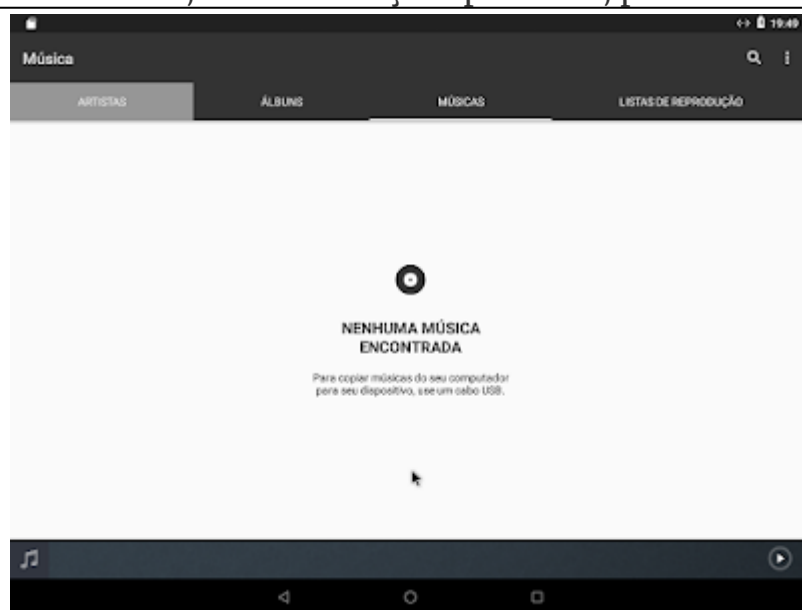
Galeria AOSP no Androix x86 8.1 Oreo R1.

Ao utilizar o *app* para selecionar um plano de fundo, uma surpresa: ele ainda mantém resquícios do Holo, tema do Android 4.x, destoando completamente do resto do sistema.

Pelo menos conseguiu reproduzir até mesmo um vídeo no formato *Matroska* (MKV) sem grandes problemas.

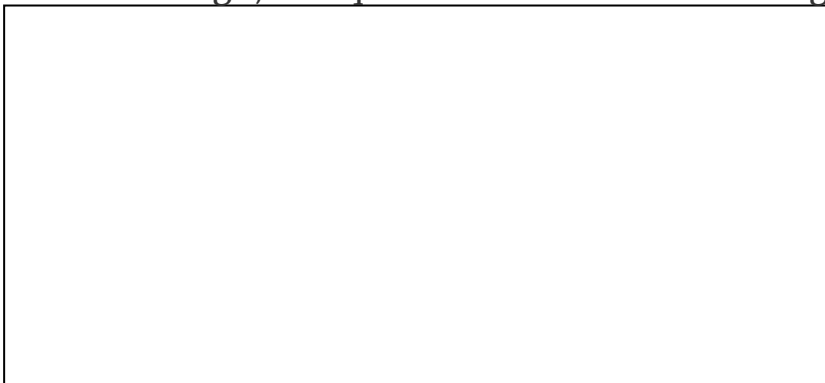
Gmail

Desde o Nougat, o Gmail é o *app* de e-mail padrão do Android x86 e, em teste com uma conta corporativa diferente da Google, funcionou sem grandes problemas.

Gmail 9 no Android x86 8.1 Oreo R1.

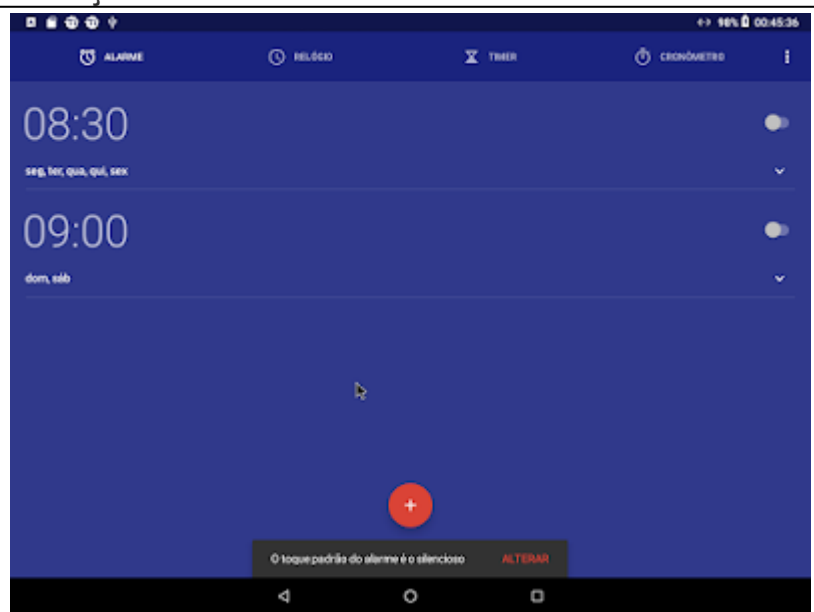
Música

Fora o ícone, sem mudanças aparentes, permanecendo na versão 3.0.

Música Lineage OS 3 no Android x86 8.1 Oreo R1.

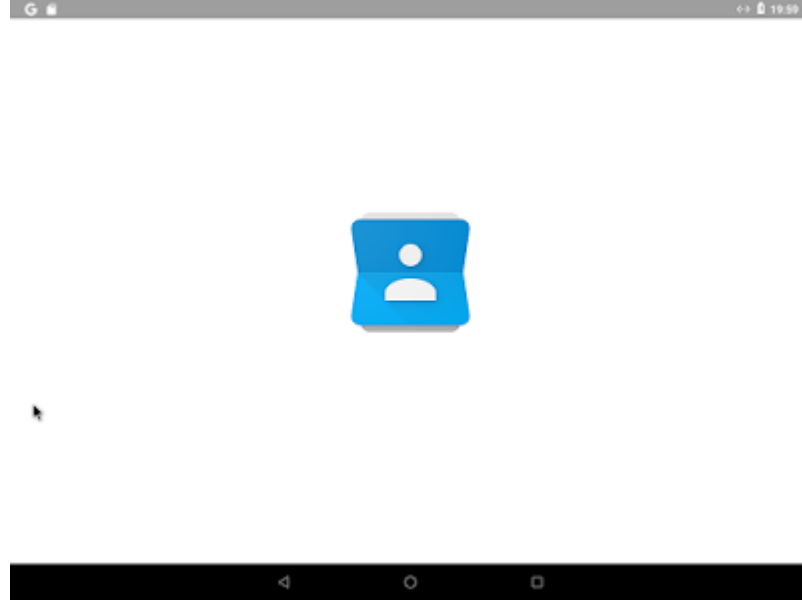
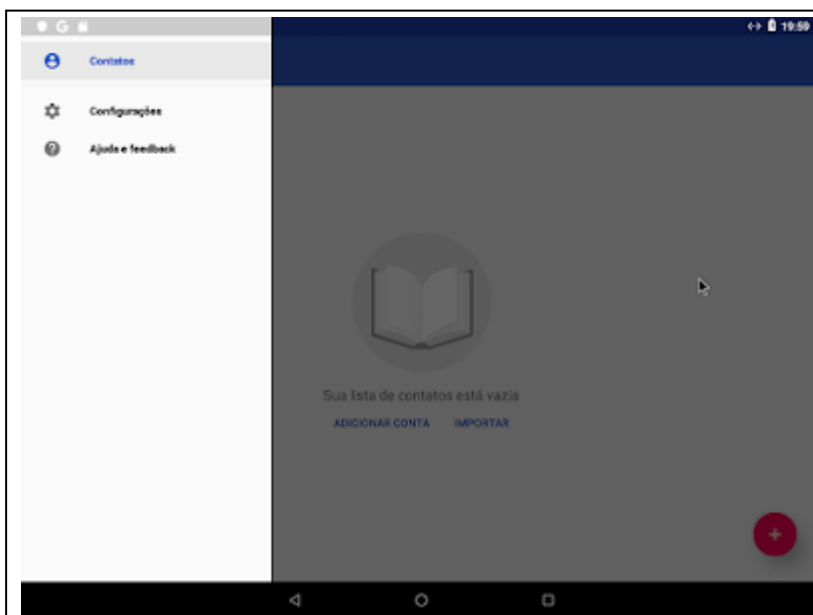
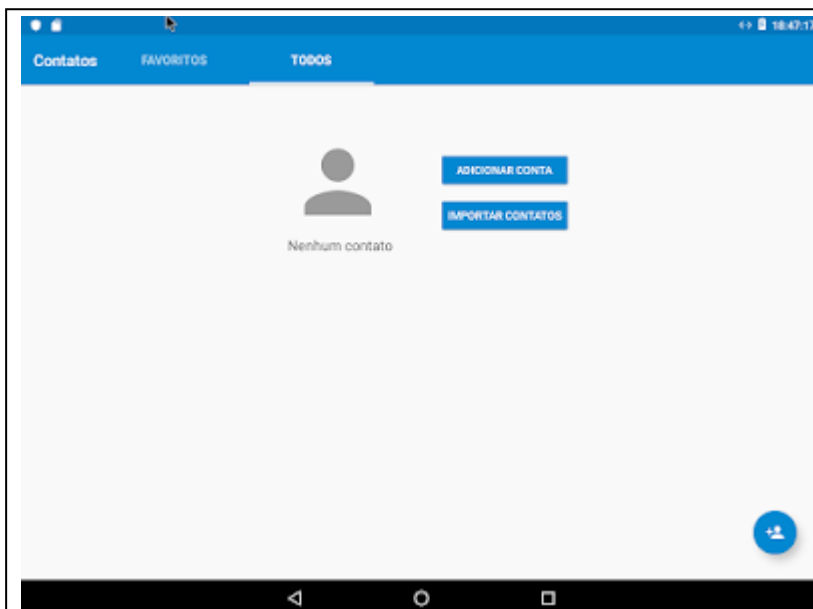
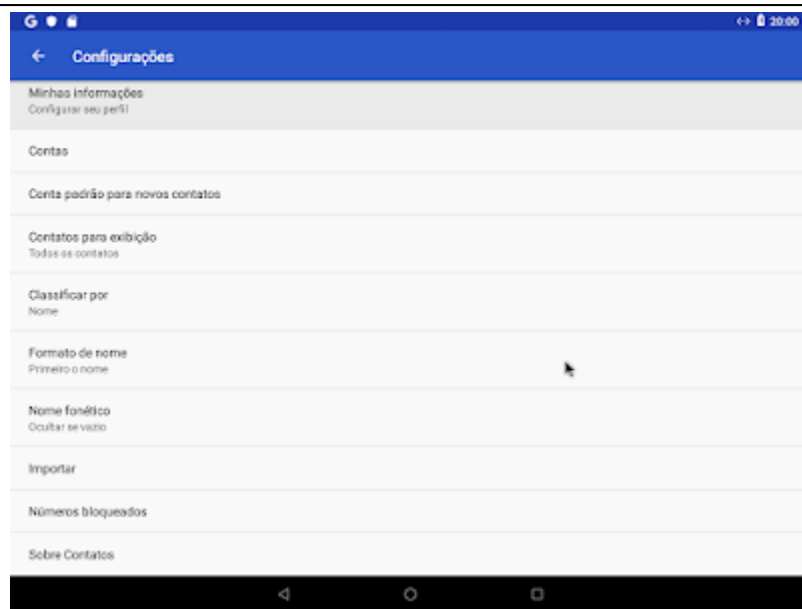
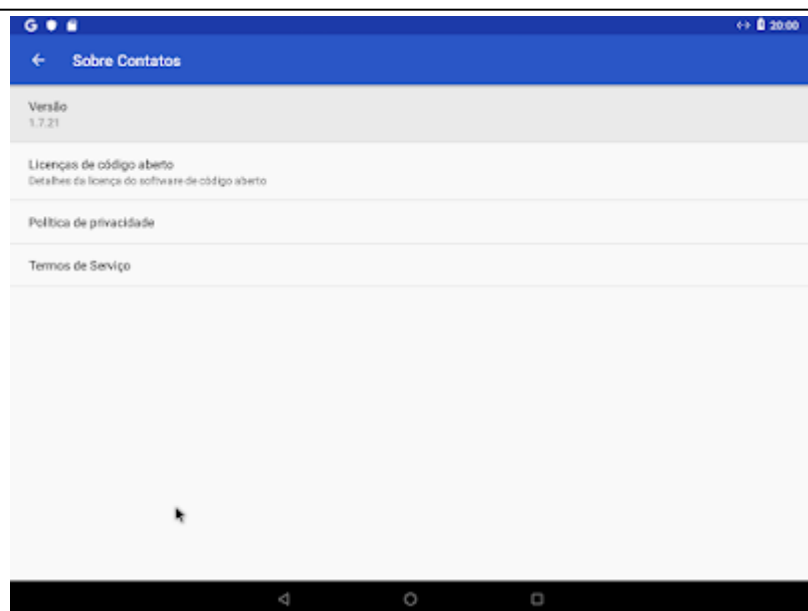
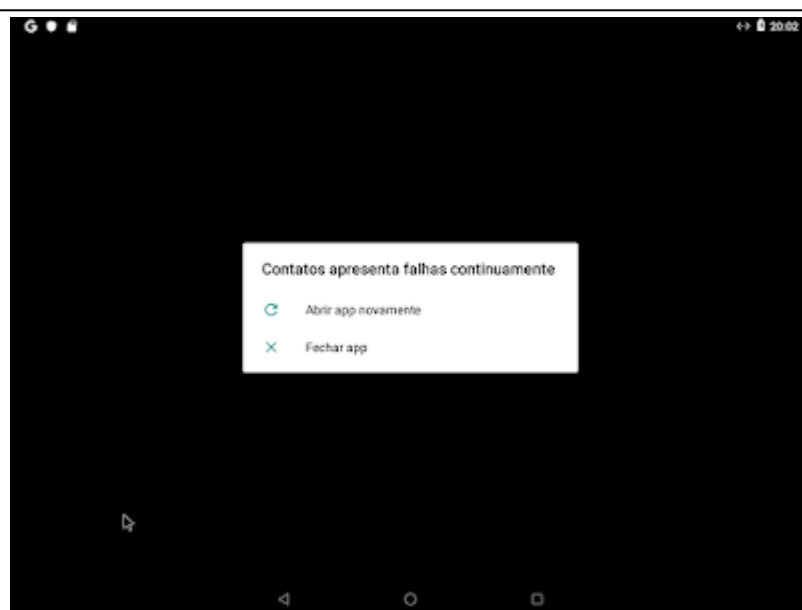
Notas

Também não sofreu quaisquer mudanças, permanecendo não adaptado ao Material Design, fora que ainda está somente em inglês.

Notas AOSP no Android x86 8.1 Oreo R1.

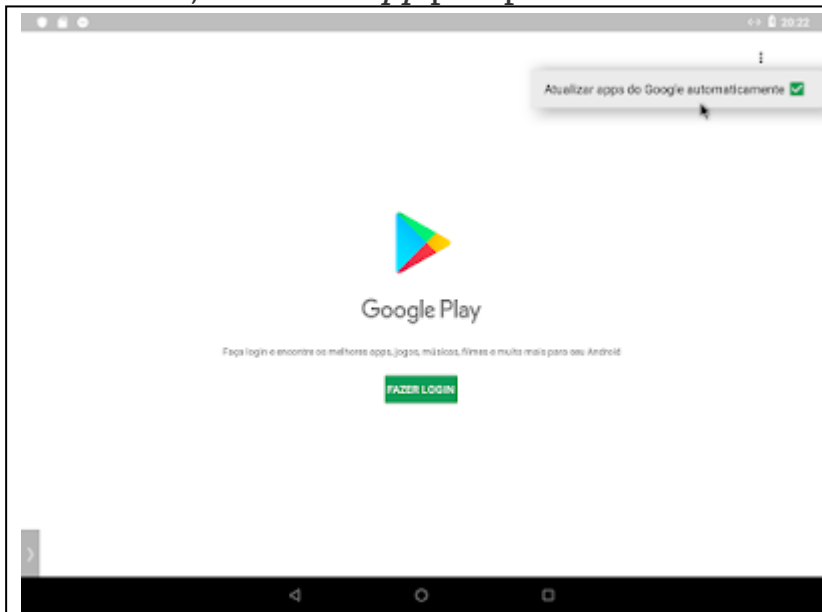
Relógio

Este sofreu algumas modificações em relação ao Nougat, ganhando descrição nos ícones do menu.

Relógio AOSP no Android x86 8.1 Oreo R1.

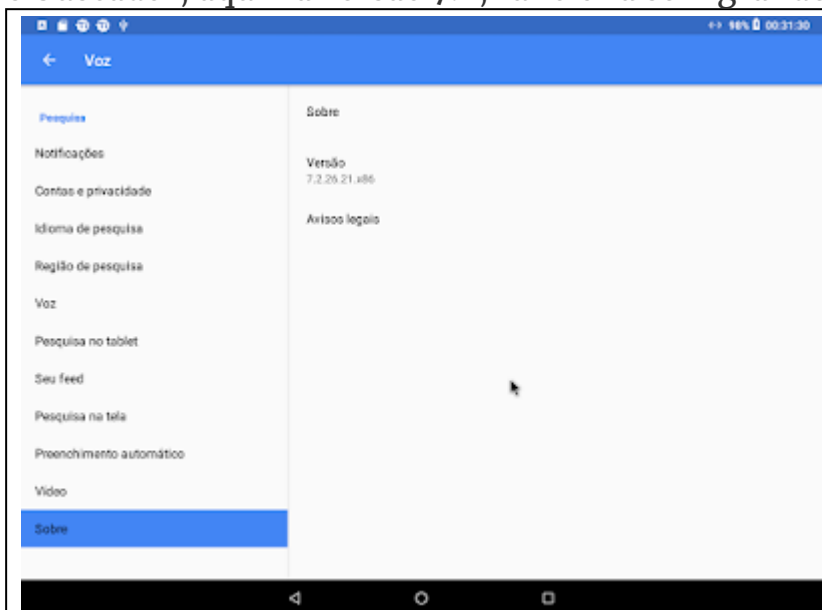
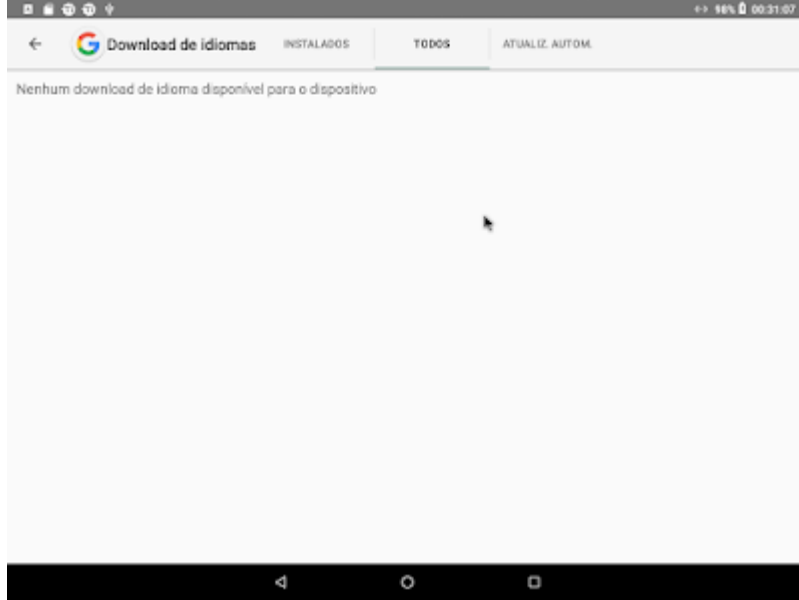
Contatos

Surpreendentemente, foi o que recebeu mais mudanças, que serão detalhadas a seguir.

De cara, ganhou um *splash* de abertura (exclusivo dos *apps* da Google - aqui, sem a identificação da empresa).

De cara, percebe-se o redesenho, em consonância com parte do design do Oreo, incorporando até um menu hambúrger.

Este é a tela inicial do *app* no Android x86 7.1.2 Nougat R1 (mais próximo de duas versões anteriores).

Nas configurações fica mais perceptivo o quanto o *app* foi aprimorado.

Tela Sobre.

Contudo, ao clicar na opção Números bloqueados, o *app* trava (*bug* que veio das compilações preliminares, também presente na primeira estável).

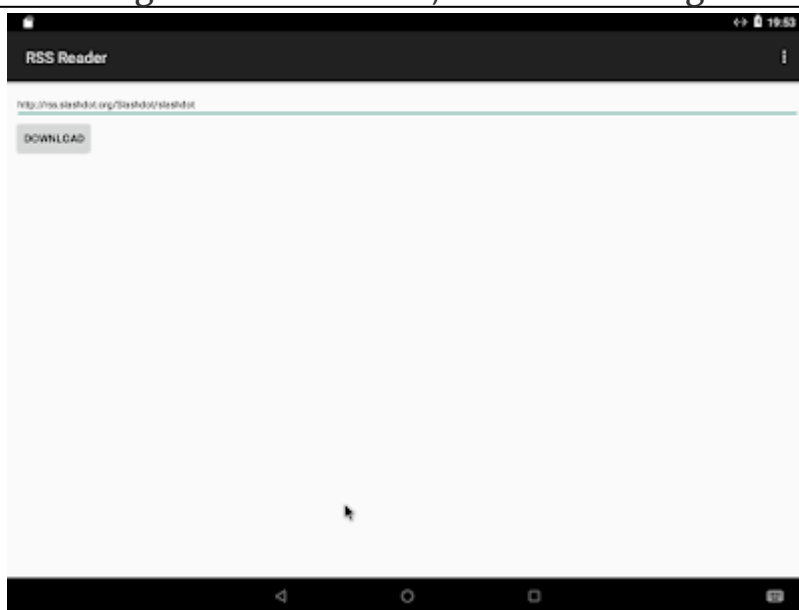
Play Store

Nos últimos tempos, a loja de aplicativos da Google (ainda imbatível em questão de gerenciamento de aplicativos - mais profundo que qualquer concorrente, inclusive de outros sistemas - por baixar somente o necessário e ter um método de instalação próprio) ganhou esta tela de boas vindas, ao abrir o *app* pela primeira vez:

Tela de boas vindas do Google Play Store 13.

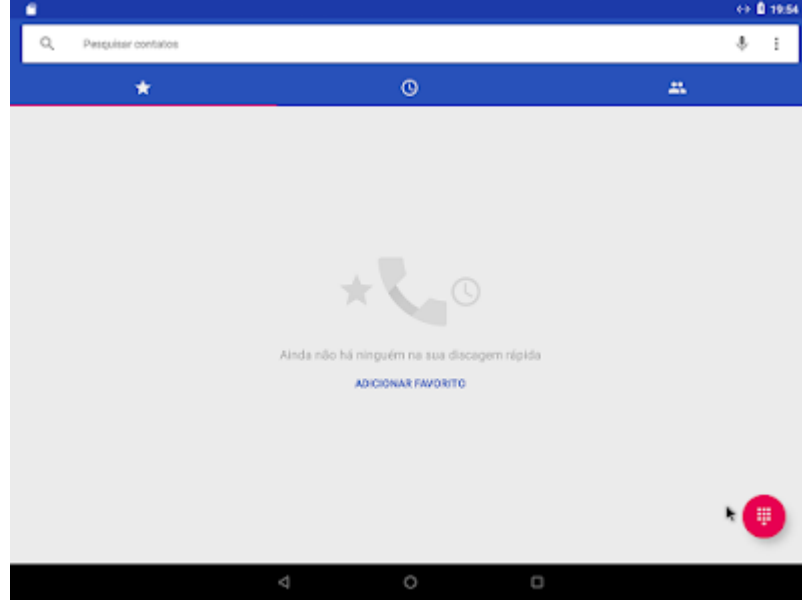
Pesquisa da Google

O buscador, aqui na versão 7.2, funciona sem grandes problemas.

Pesquisa da Google no Android x86 8.1 Oreo R1.

O único detalhe é o Reconhecimento de fala off-line, que além de não estar preparado para o uso com o Ethernet, permanece com o design do Android 4.x.x.

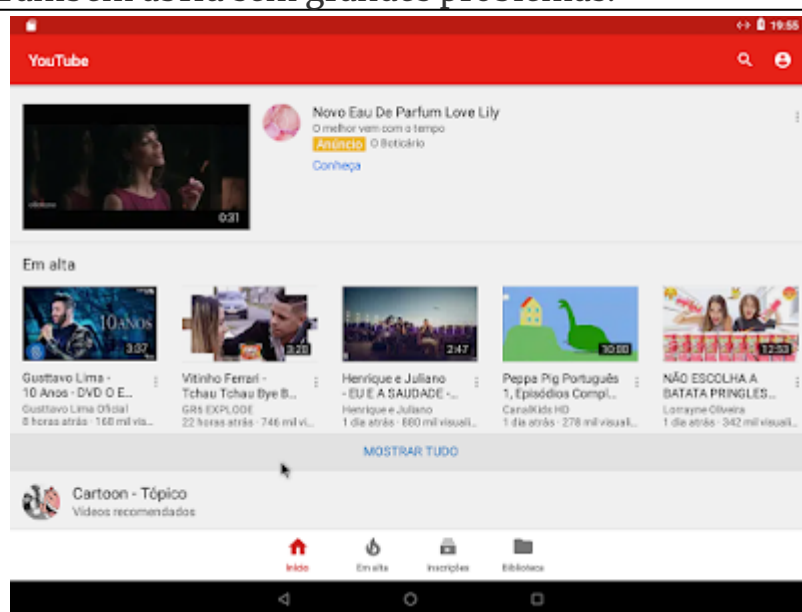
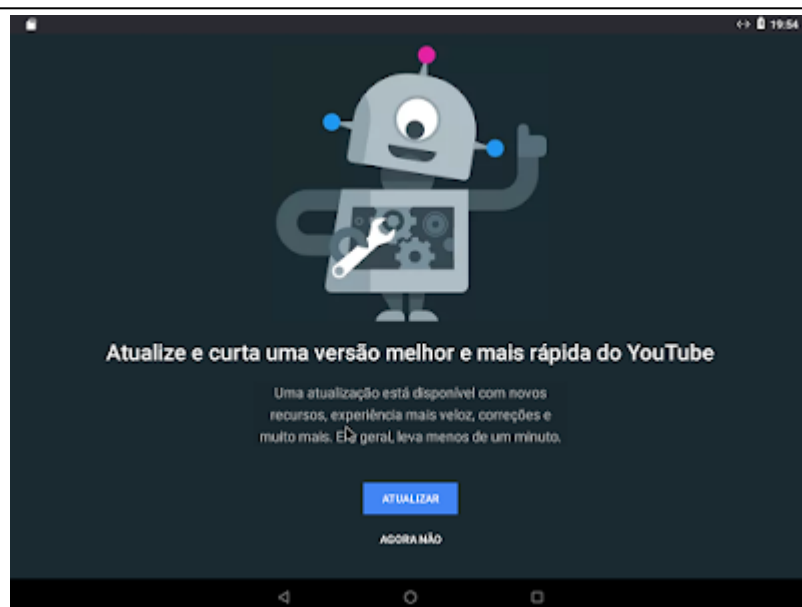
RSS Reader

Não teve grandes novidades, mas abriu sem grandes problemas.

Assim como o Dev Tools, o *app* teve o fundo alterado para a cor branca - antes era todo escuro.

Telefone

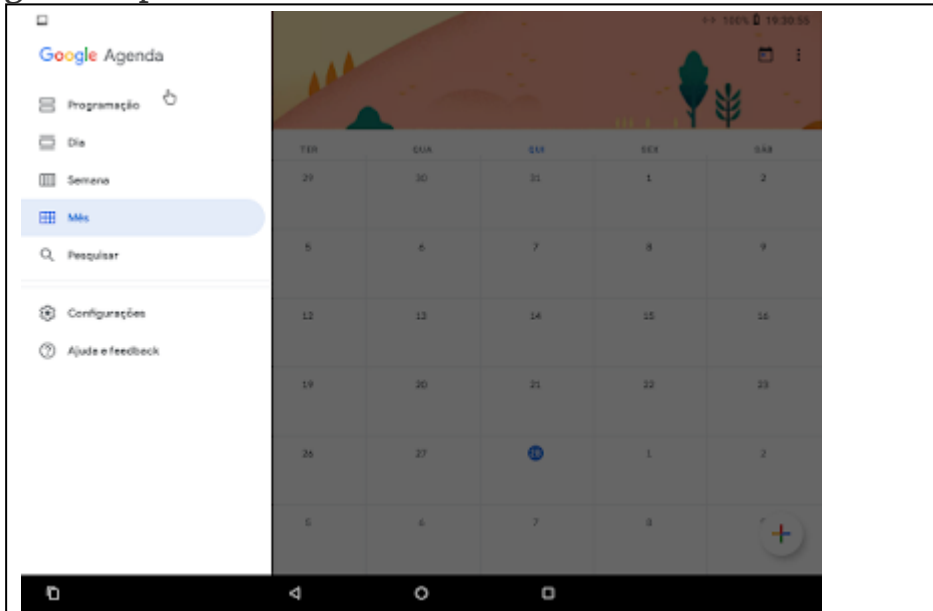
Diferente do Contatos, o discador não recebeu praticamente nenhuma mudança.

Telefone AOSP no Android x86 8.1 Oreo R1.

YouTube

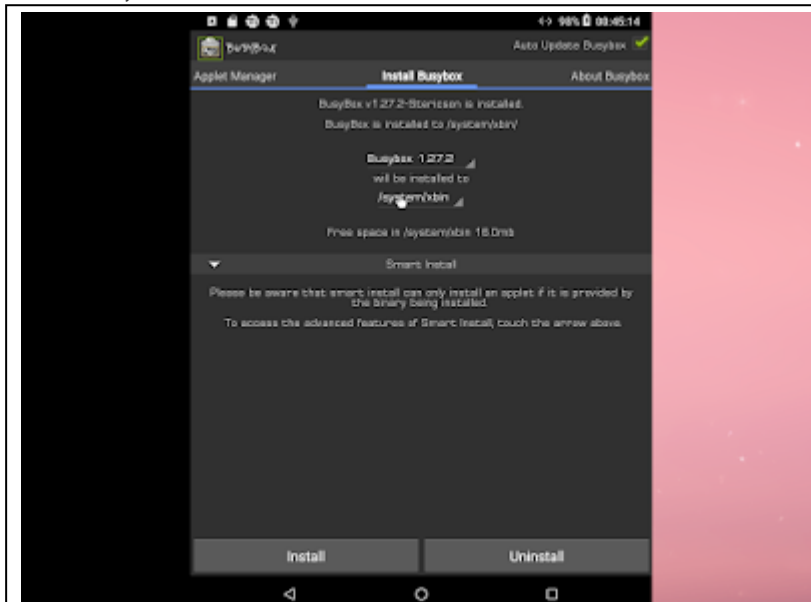
Também abriu sem grandes problemas.

YouTube 12 no Android x86 8.1 Oreo R1.

Curiosidade: o aviso de aplicativo desatualizado ficou mais chamativo e criativo.

Outros aplicativos

Agenda do Google

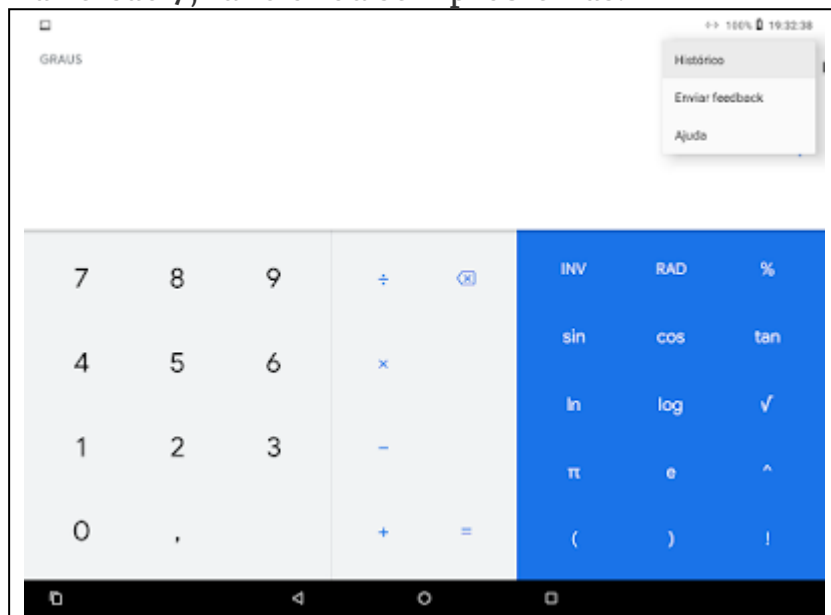
O calendário da empresa de Mountain View, aqui na versão 6, abriu sem grandes problemas.

Mas também não escapou dos problemas de autenticação.

Busybox

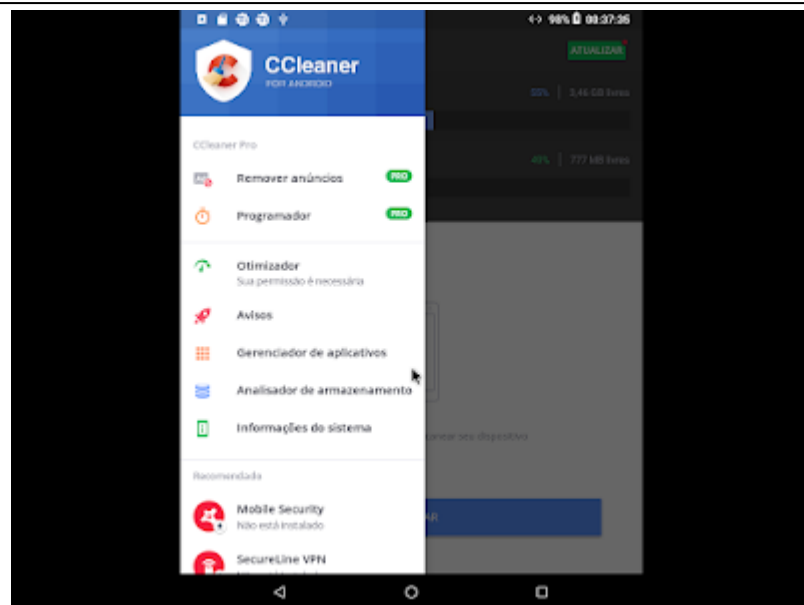
O aplicativo, que gerencia os comandos instalados no sistema, desenvolvido por [Stericson](), também funcionou sem grandes problemas (importante ressaltar que a compilação vem com a versão 1.22.1 do recurso).

Contudo, o *app* ainda não está adaptado para o modo Paisagem (aproveitando, a primeira compilação estável do sistema ajusta por

padrão a tela no modo retrato corretamente, sem dores de cabeça).

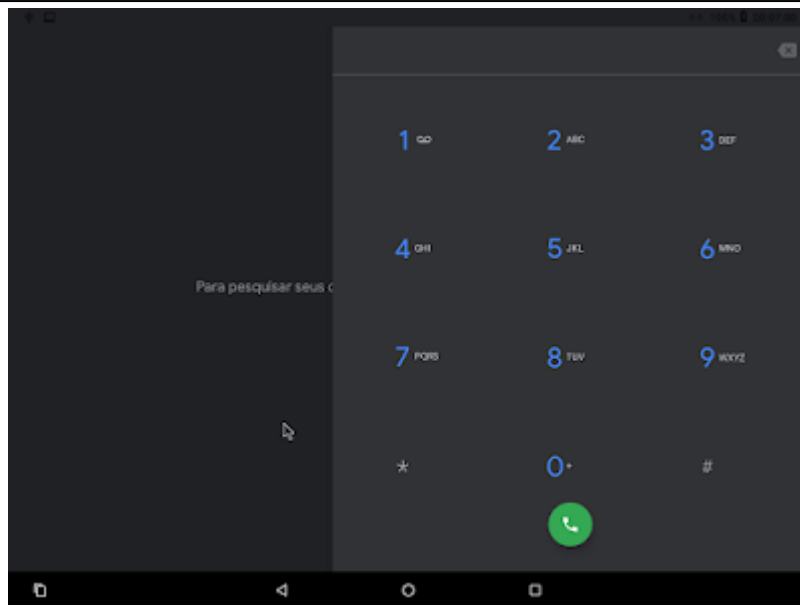
Calculadora do Google

Na versão 7, funcionou sem problemas.

Calculadora Google no Android x86 8.1 Oreo R1.

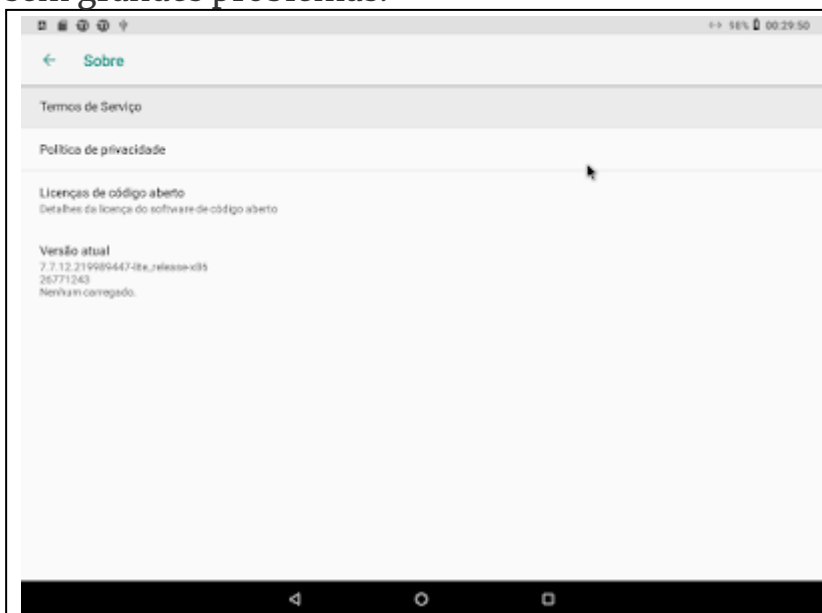
CCleaner

Neste meio tempo, a desenvolvedora do app, Piriform, foi adquirida pela Avast Software, e as influências da empresa Checa são evidentes ao "*linkar*" seus *apps* no menu hambúrger, do mesmo jeito que está no antivírus dela.

Desde a análise do Marshmallow, o *app* permanece não suportando o modo paisagem.

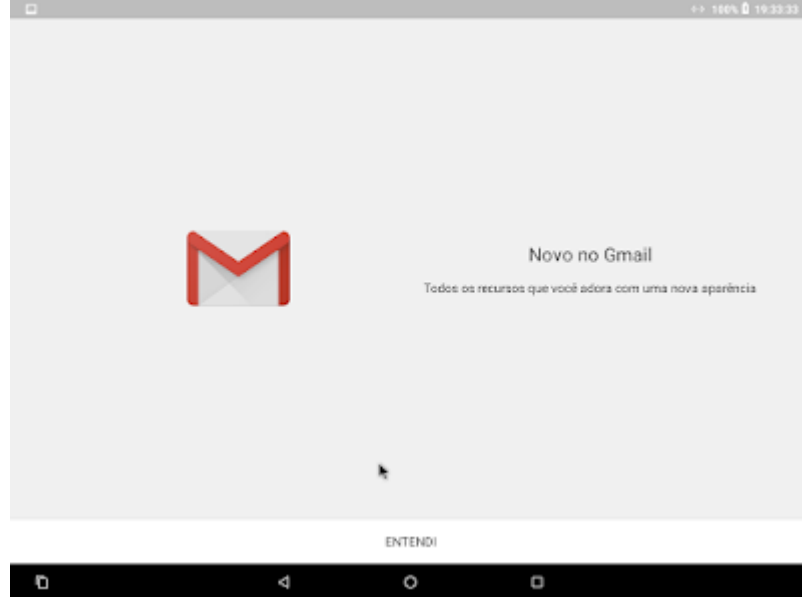
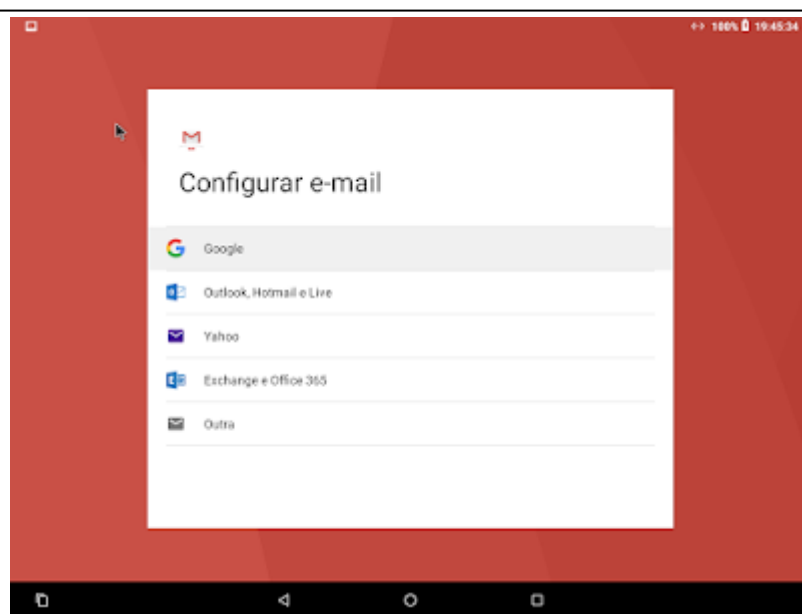
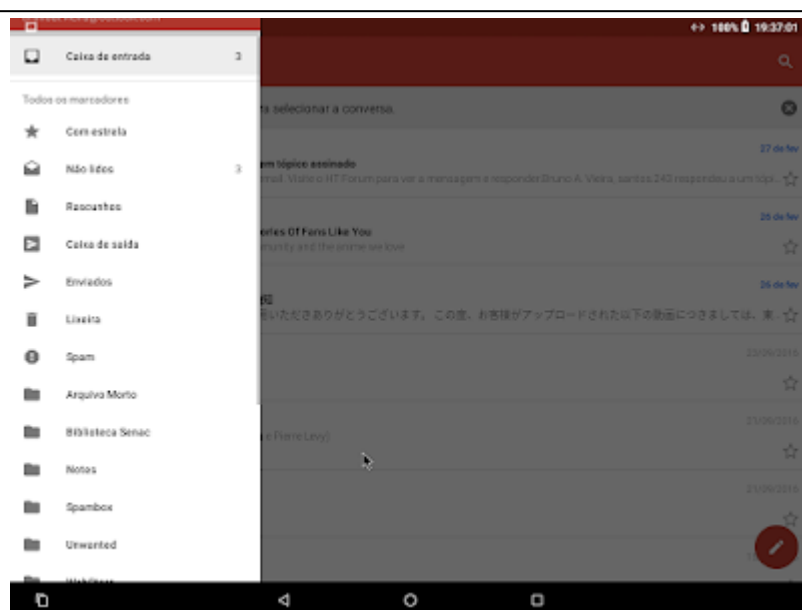
Telefone Google

Sem muitas diferenças em relação ao aplicativo AOSP, funcionou sem problemas.

A versão testada permitia utilizar o tema escuro, independente do sistema.

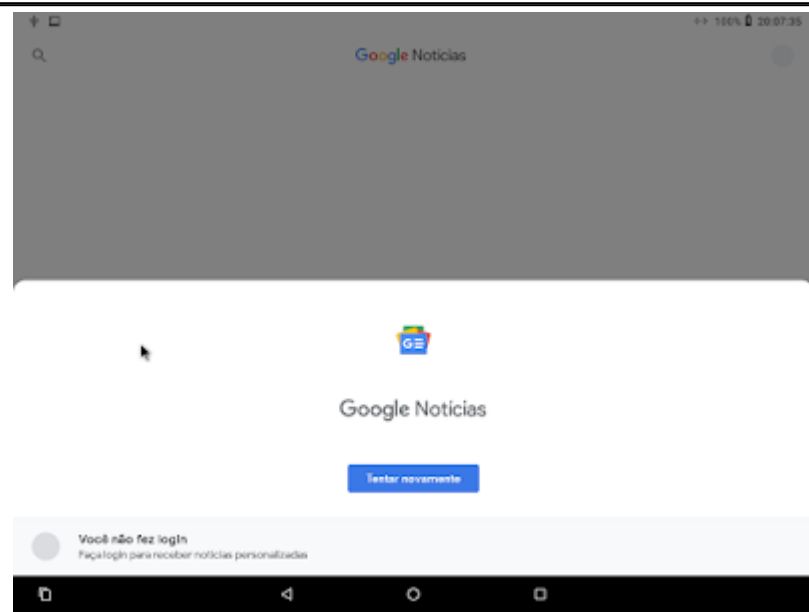
Gboard Lite

A versão enxuta (ou deveria ser, já que o tamanho do APK não diferente tanto do original, fora que ainda oferece a maior parte dos recursos) abriu sem grandes problemas.

Gboard Lite no Android x86 8.1 Oreo R1.

Gmail Go

Uma alternativa para quem utiliza a versão 8.1 ou superior do Android é a versão simplificada do gerenciador de e-mails da Google que, no geral, funcionou sem grandes problemas.

A tela de boas vindas é um pouco modificada em relação ao *app* padrão.

O assistente de configuração, por sua vez, é idêntico ao do Gmail padrão.

A tela principal em si, lembra um pouco as versões mais antigas do *app* normal.

Google Notícias

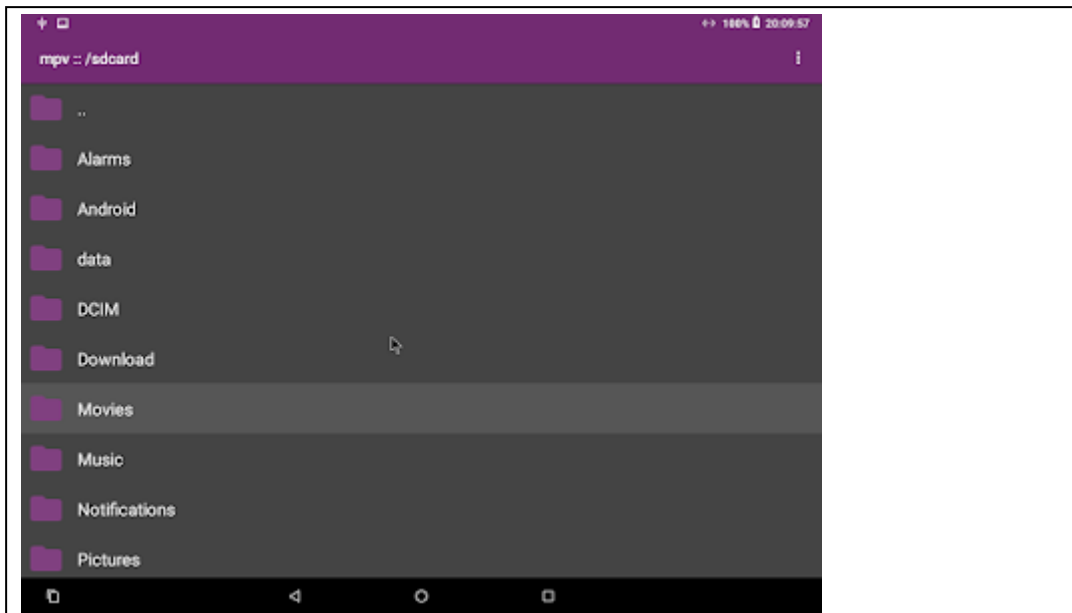
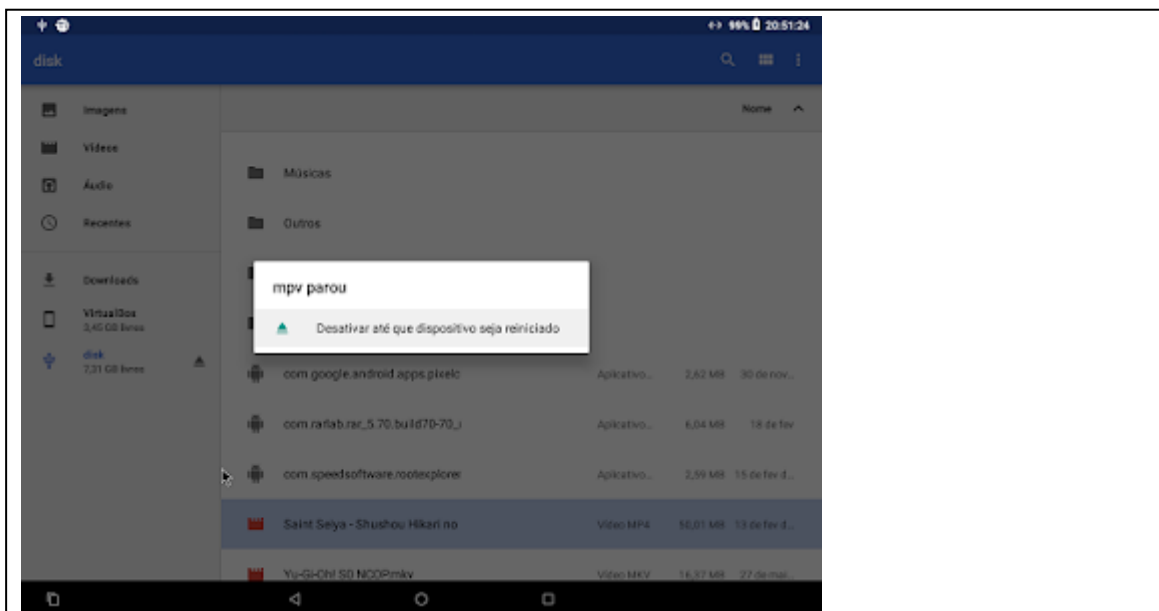
Sucessor do Google Play Banca e substituto oficial do Notícias e Clima, o aplicativo abriu sem grandes problemas; entretanto, ao contrário do antigo *app*, este exige autenticação para poder começar, o que. por sinal, é diferente de seus concorrentes, como o Microsoft Notícias.

Google Notícias 5.8 no Android x86 8.1 Oreo R1.

FFmpegui

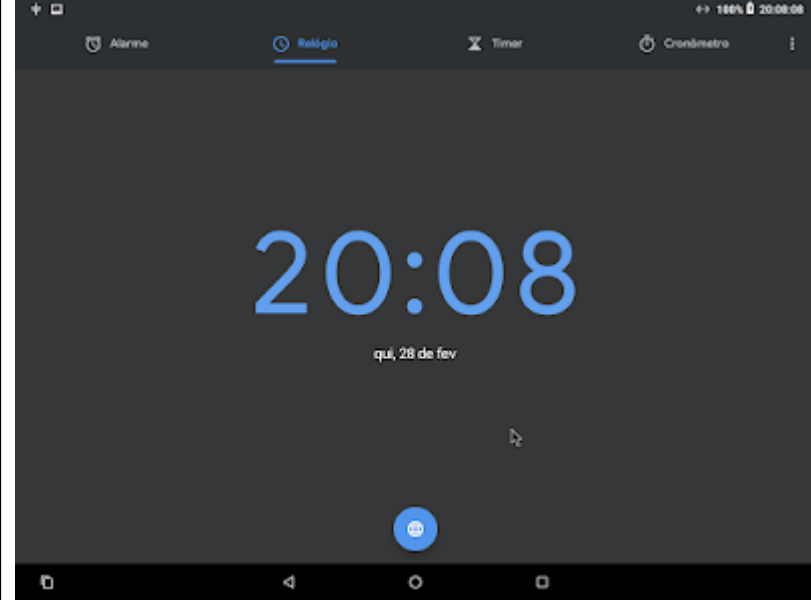
Uma alternativa robusta para o Android, que permite ter acesso à maioria dos recursos do famoso gerenciador de mídia de código aberto, abriu sem grandes problemas.

Ele permite o download de binários personalizados para a arquitetura 32 bits, ao pesquisar pelas opções.

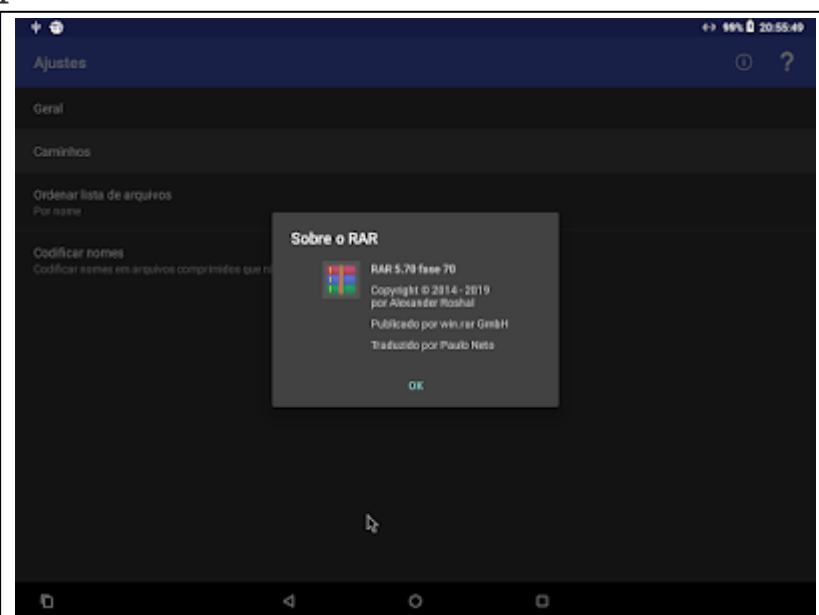
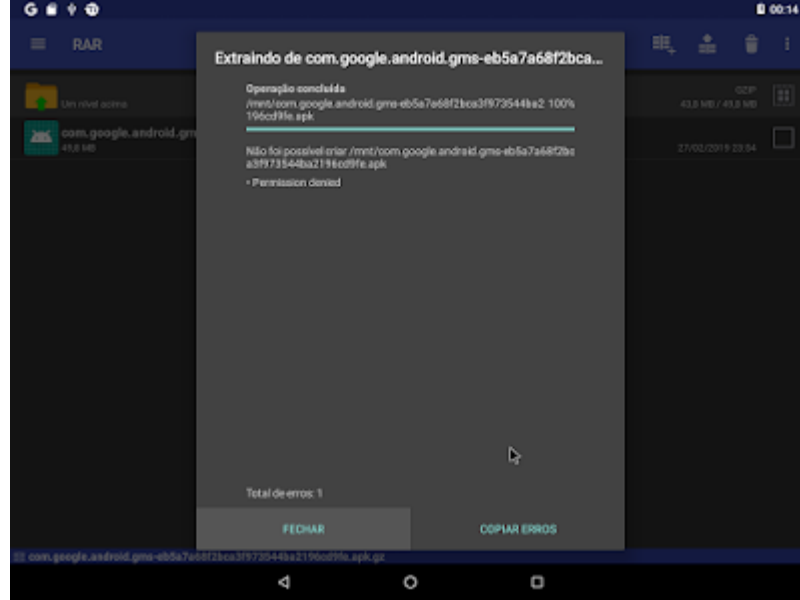
mpv

É um dos reprodutores de mídia mais clássicos de distribuições GNU/Linux, mas que ainda está em estágio *alpha* no Android (até a data de publicação original desta análise).

O *app* até abriu e permitiu a navegação, como pode ser visto acima.

Entretanto, não conseguiu reproduzir nada, simplesmente dando *crash*.

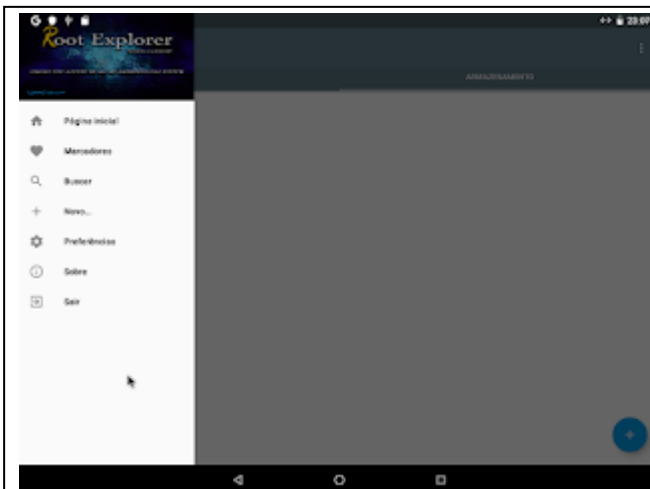
Relógio do Google

O aplicativo, aqui em sua versão 6, também abriu sem problemas.

Relógio do Google no Android x86 8.1 Oreo R1.

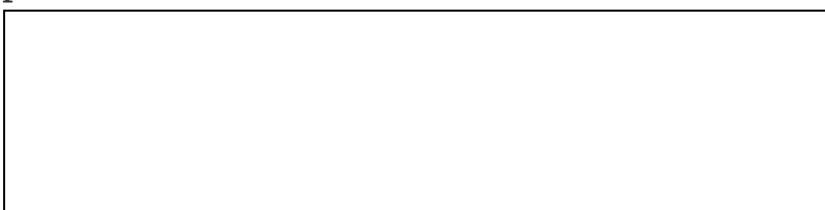
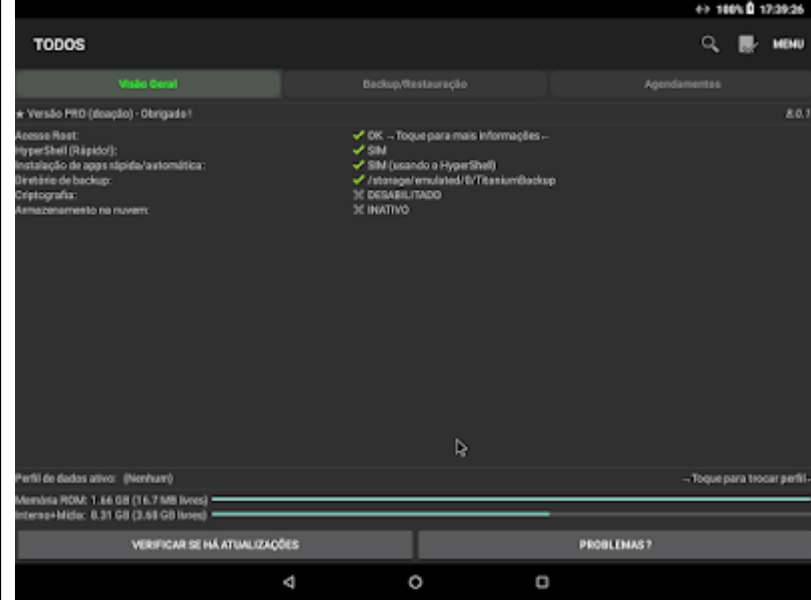
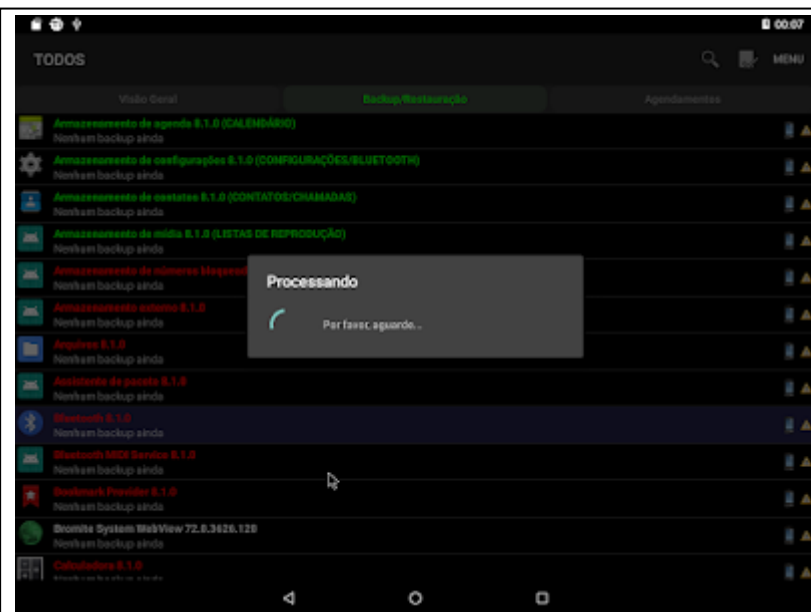
RAR

O aplicativo abriu e visualizou arquivos compactados sem grandes problemas.

RAR 5 *build* 70 no Android x86 8.1 Oreo R1.

Mas, ao tentar fazer operações dentro de arquivos do sistema, ele não conseguiu e sequer exige acesso super usuário.

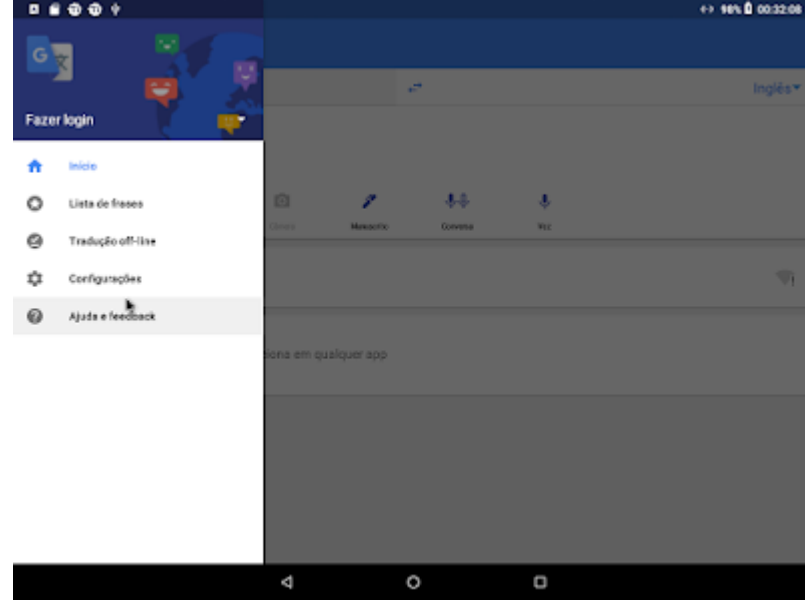
Root Explorer

O *app* desenvolvido pela Speed Software é uma das alternativas, caso se queira acessar as pastas do sistema por modo gráfico, já que o gerenciador de arquivos nativo não suporta esta opção.

A versão 4, que redesenhou o design para os padrões do Material Design do Lollipop, funcionou sem grandes problemas, embora tenha demorado para reconhecer o acesso *root* e lentidão para abrir a guia do caminho raiz do sistema.

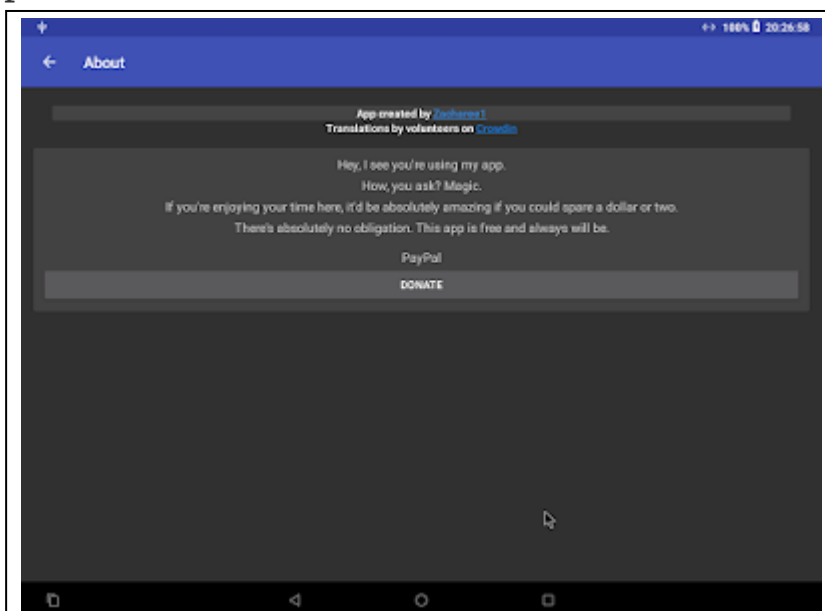
Titanium Backup

Sem grandes mudanças na aparência, a versão 8 abriu sem grandes problemas.

Titanium Backup 8 no Android x86 8.1 Oreo R1.

Entretanto, ele não conseguiu gerenciar os aplicativos do sistema, mesmo no modo leitura-escrita (algo que o *app* fez na compilação R1 do Nougat).

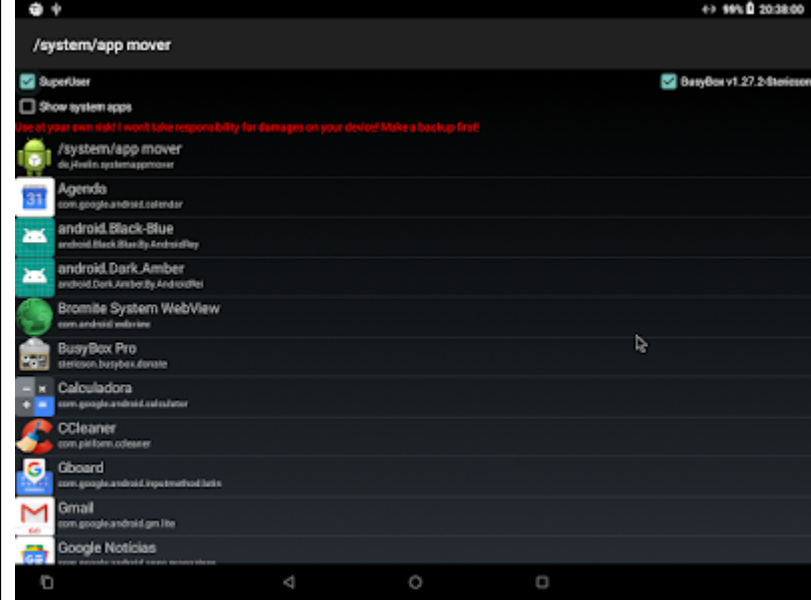
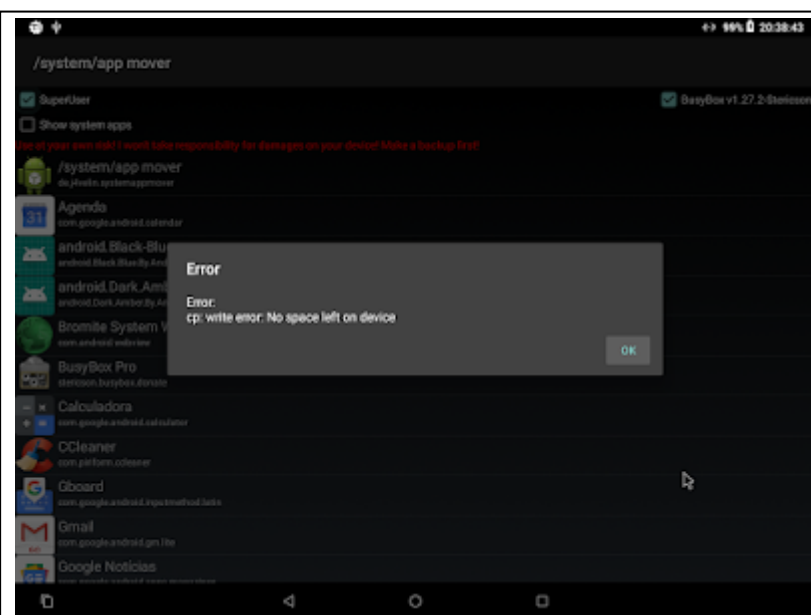
Tradutor do Google

Também abriu sem problemas.

Contudo, ele também não está adaptado para trabalhar com conexão cabeada, permitindo baixar os dados offline somente por Wi-Fi.

System UI Tuner

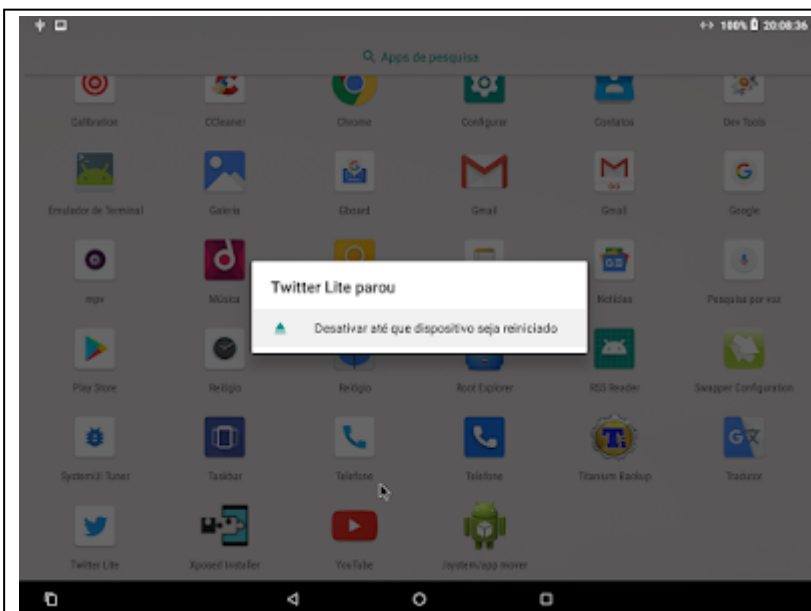
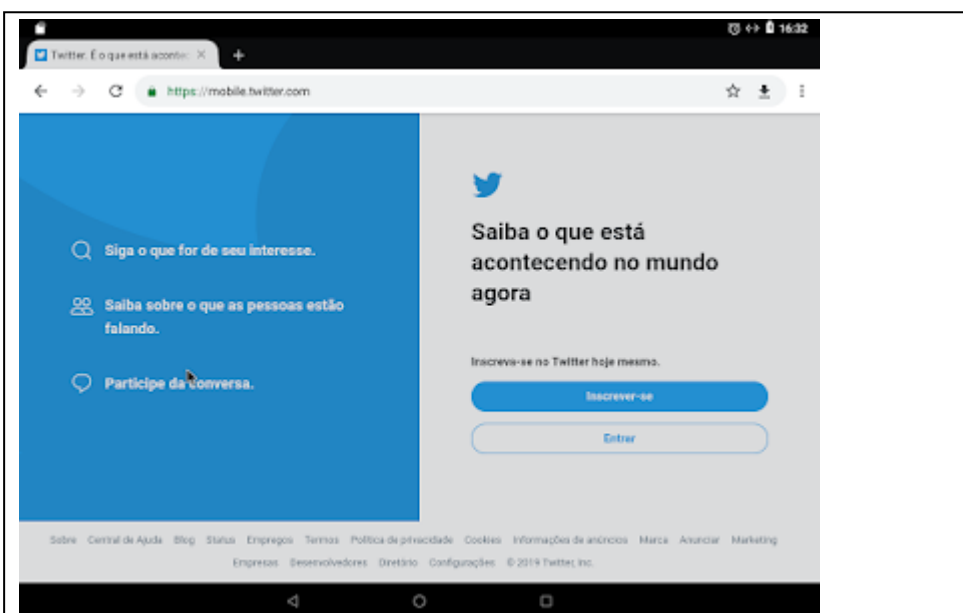
Uma alternativa ao Sintonizador System UI do sistema (especialmente em aparelhos da Samsung, que não possuem o recurso nativamente), desenvolvida por Zachary Wander, com mais opções de personalização, sem exigir acesso *root* (embora requisite ativação de determinadas permissões via shell do ADB), funciona sem grandes problemas.

System UI Tuner no Android x86 8.1 Oreo R1.

/system/app mover

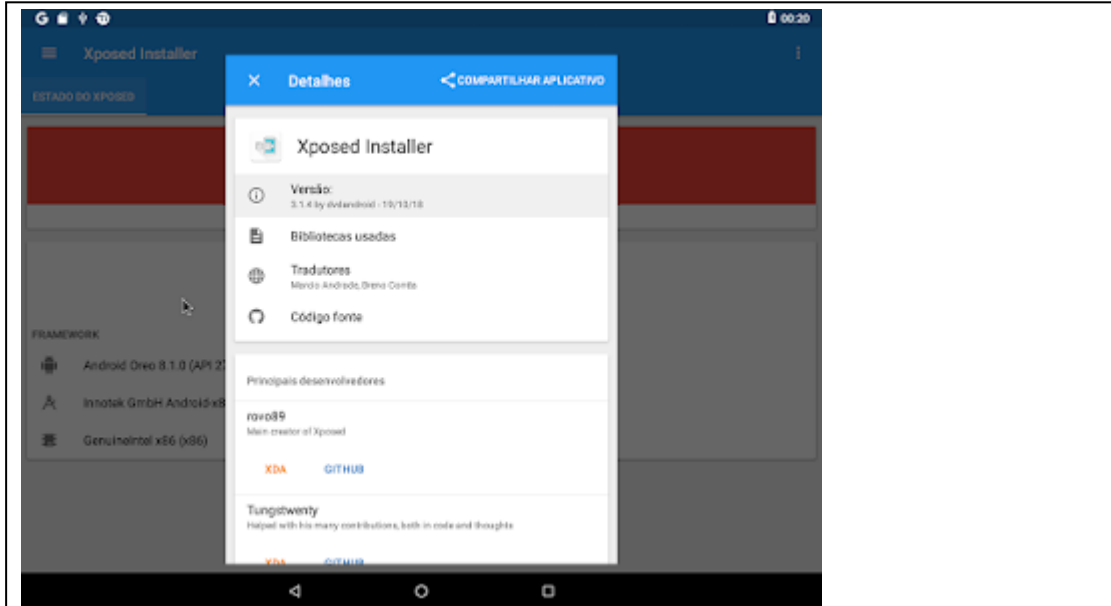
Uma boa opção para gerenciar apps abriu sem grandes problemas, mas levou um certo tempo para reconhecer o modo super usuário.

/system/app mover 1.72 no Android x86 8.1 Oreo R1.

Contudo, não conseguiu cumprir sua função, dando o erro exibido na imagem acima (o que explica o mesmo problema dado no Titanium - provavelmente é limitação do sistema).

Twitter Lite

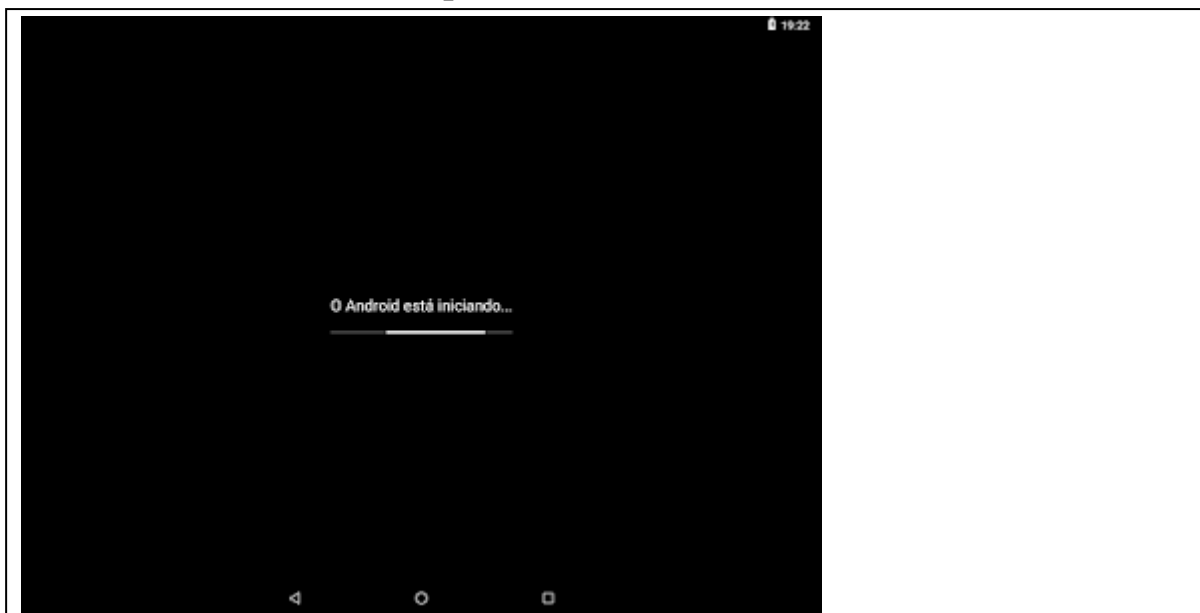
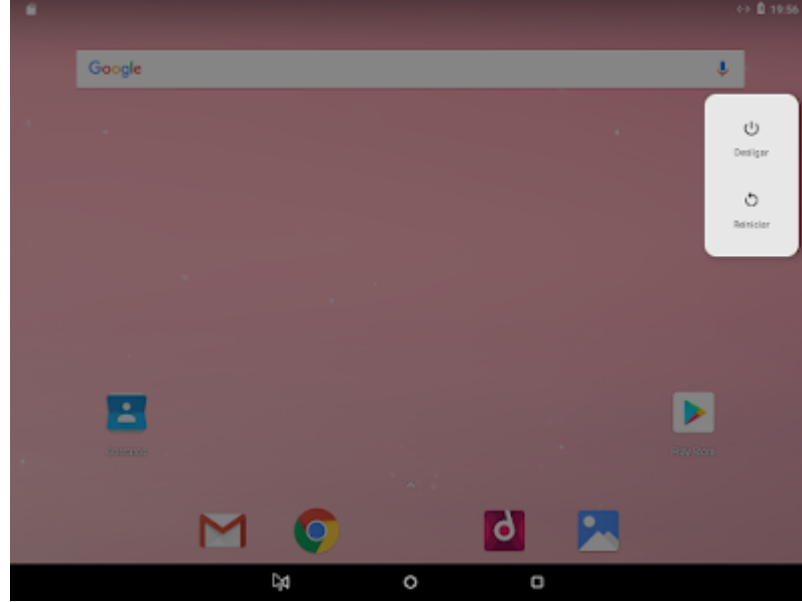
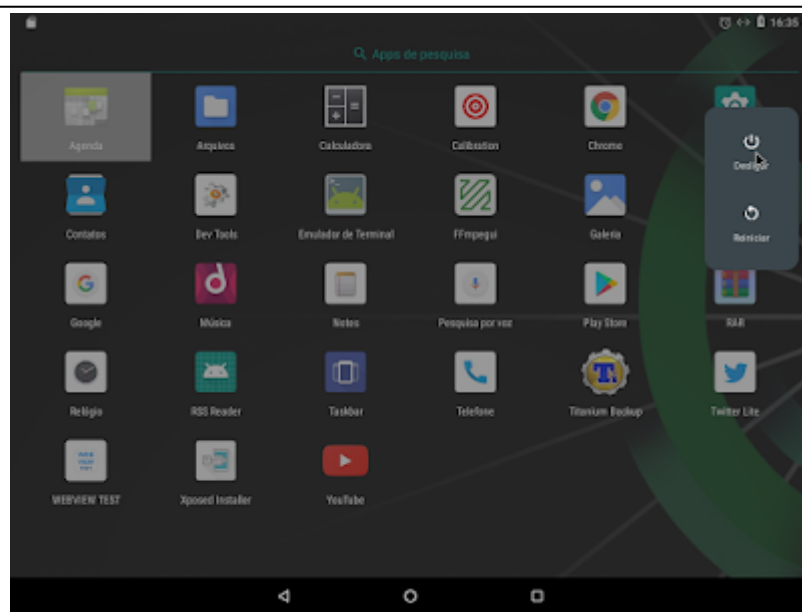
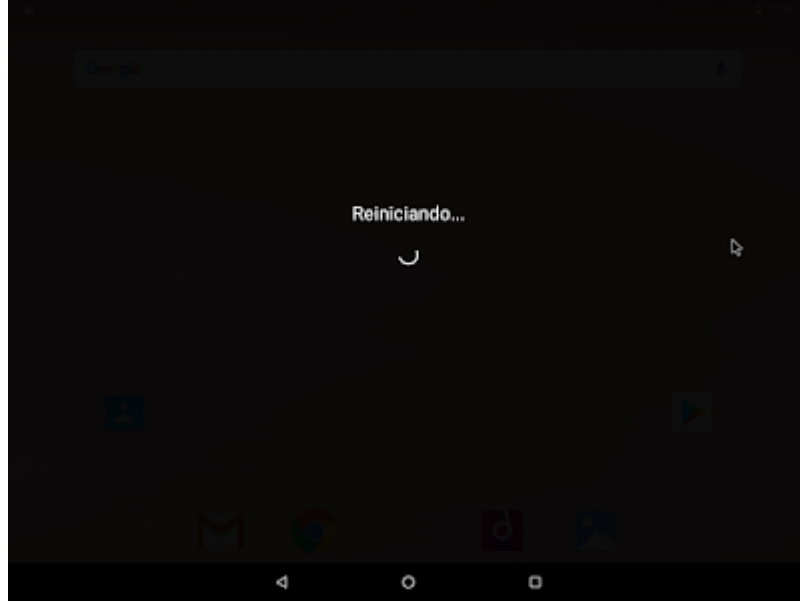
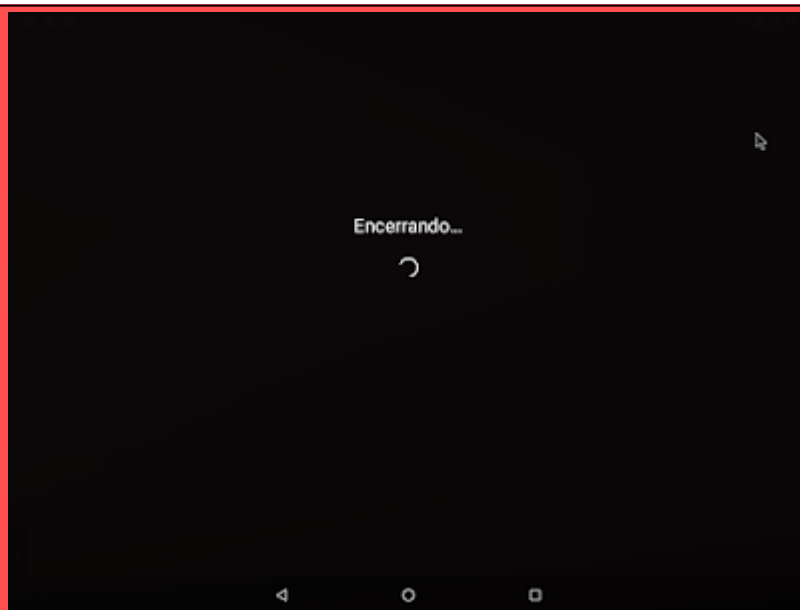
Uma alternativa à versão normal, baseado no conceito de PWA (*Progressive Web App*), também foi testado na compilação final do Oreo no Android x86.

A tela acima, retirada do testes na compilação 7.1.2 Nougat R2 do projeto, mostra como o *app* deveria se comportar em modo tablet, funcionando sem grandes problemas.

Contudo, simplesmente o Twitter Lite não funcionou no Android x86 8.1 Oreo R1, mesmo ajustando a permissão de Armazenamento.

Pelo menos, o serviço ainda pode ser acessado pelo Chrome.

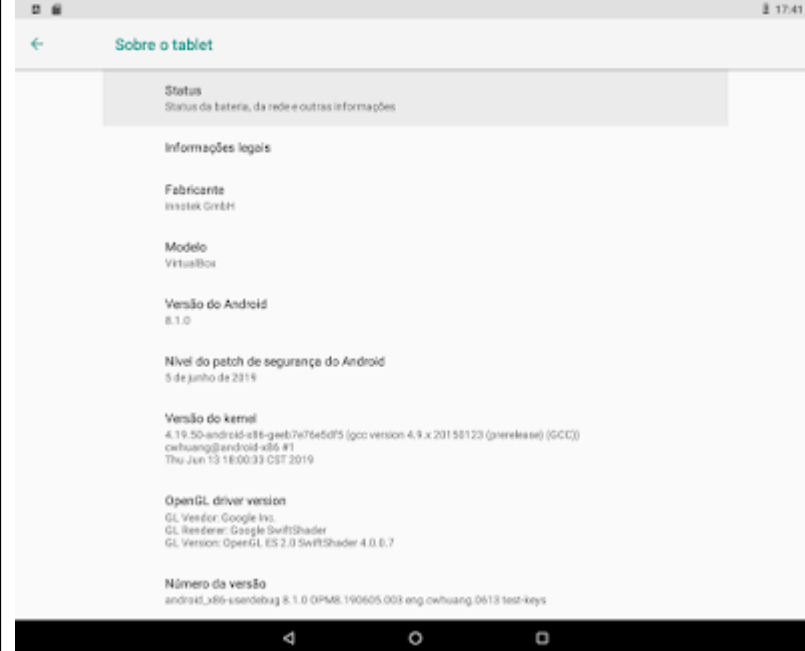
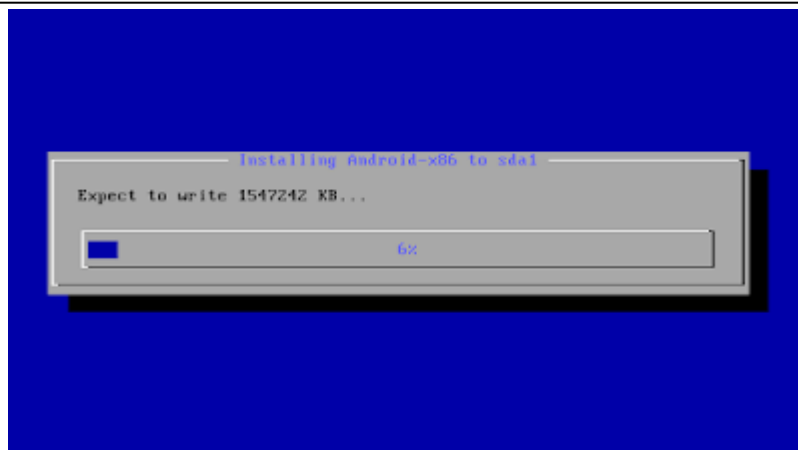
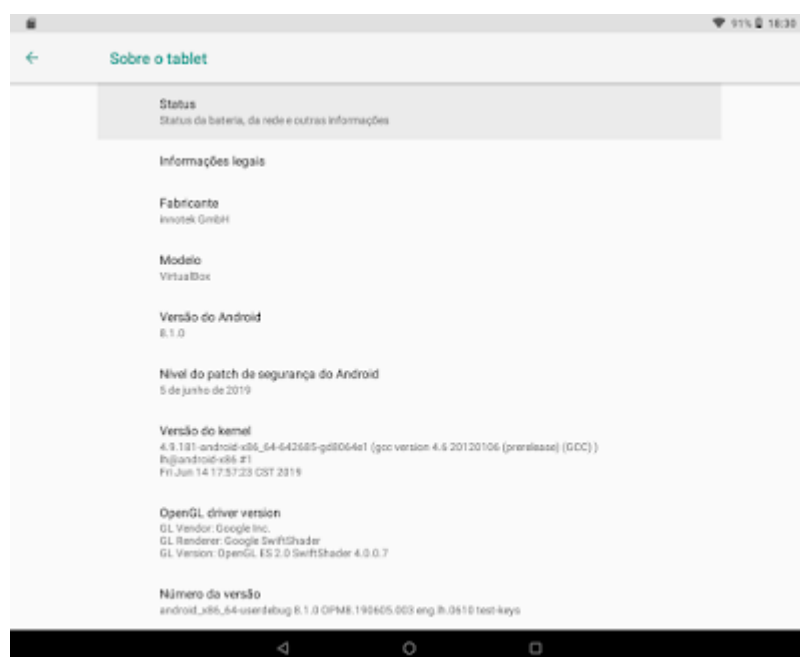
Xposed Framework

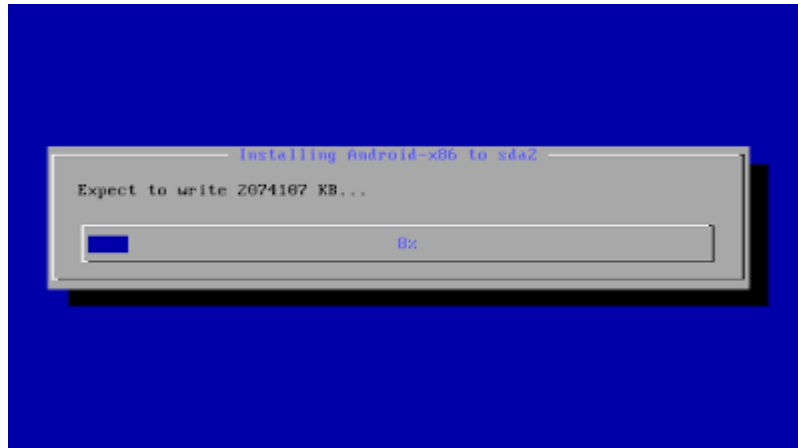
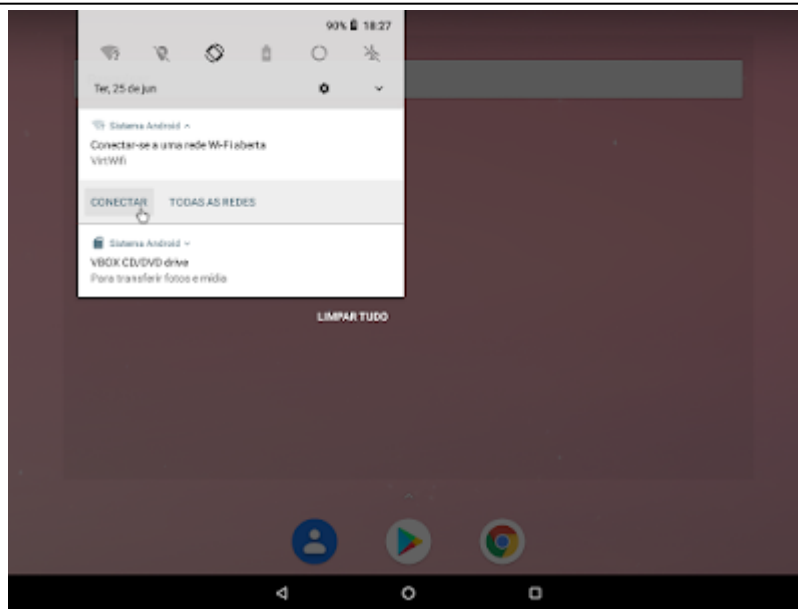
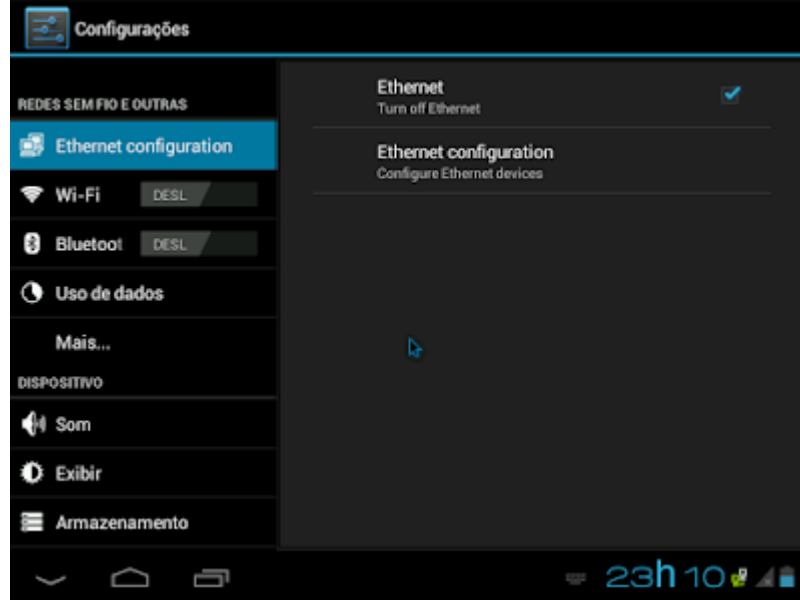
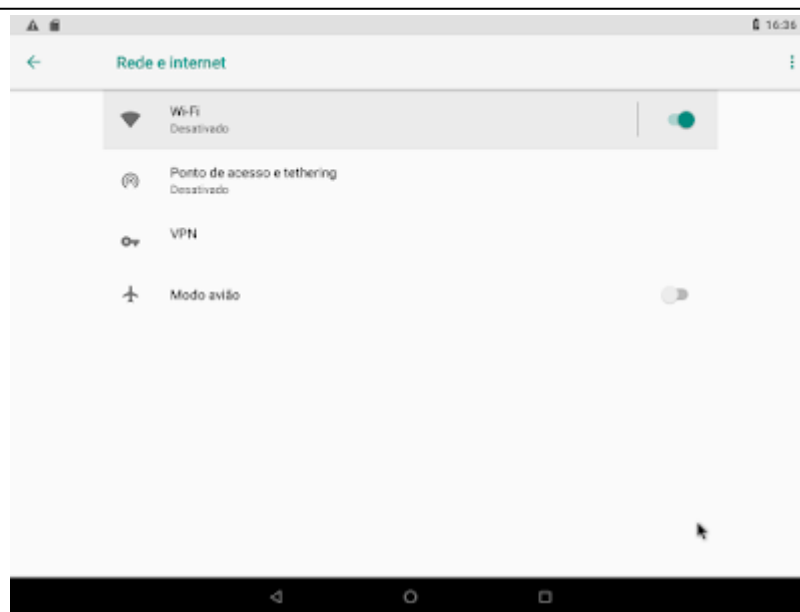
O famoso gerenciador avançado de recursos do Android abriu sem grandes problemas. Contudo, em testes, constatei que o *app* não oferece suporte às compilações do Android x86, podendo causar *bootloop*. Portanto, seu uso é desaconselhado.

Foi testada a versão 3.1, mas não permitiu o download do *framework*.

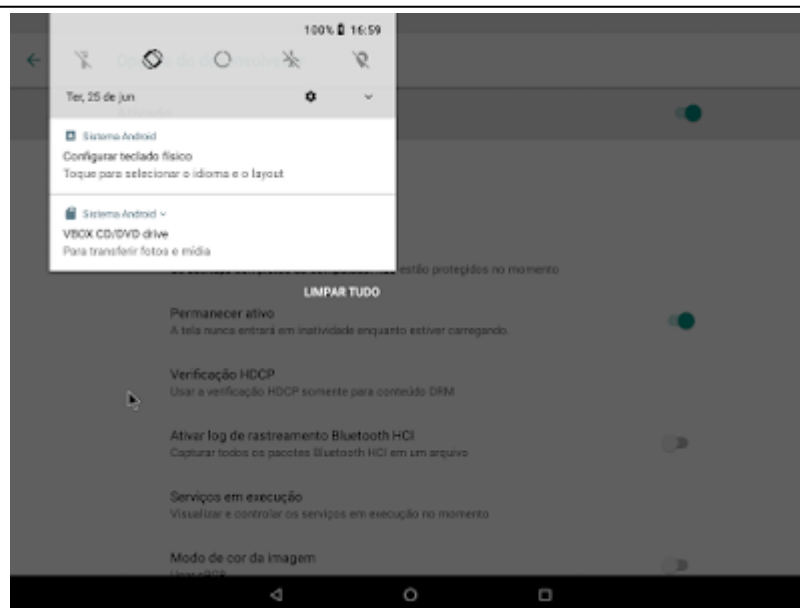
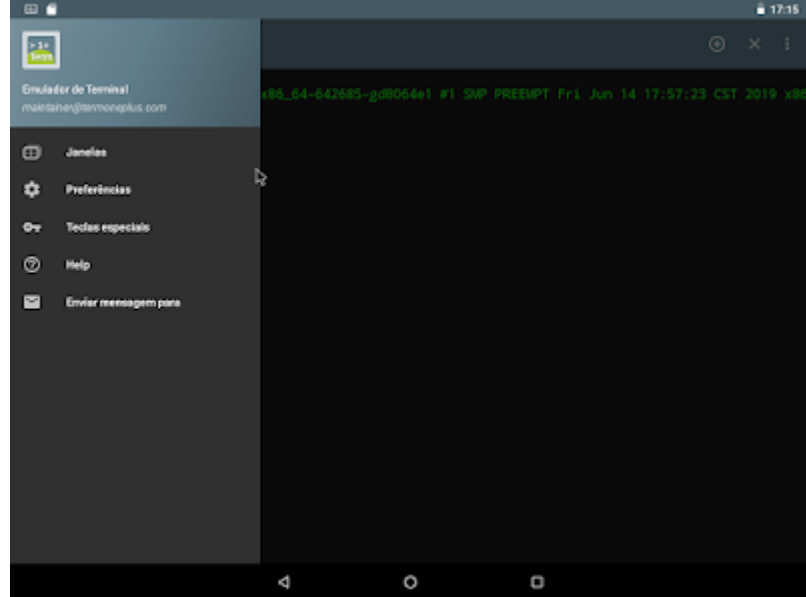
Inicialização e Encerramento

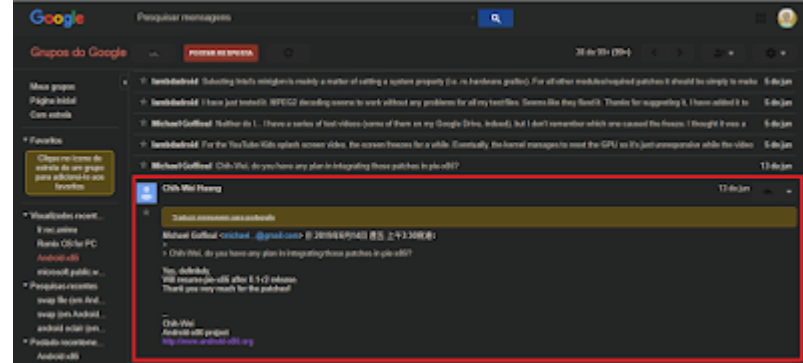
Foi outra área do sistema que recebeu novidades.

O carregamento do sistema, ao reinicializar, teve a aparência alterada no Nougat e agora exibe de maneira semelhante à Samsung após a execução do assistente inicial (isto desde o KitKat).

Finalmente, desde a versão 7.1.2 do Android, o botão Reiniciar está presente de forma nativa. E, como pode ser visto acima, o Oreo redesenhou a função (praticamente inalterado desde o ICS), deixando agora no canto (provavelmente próximo do botão físico, no caso de um *smartphone* ou *tablet* - o que pode não ser muito relevante para quem usa PC ou notebook).

Esta é uma versão alternativa dos botões, na cor escura.

Seu comportamento, ao reiniciar, é exibido como deveria ser, ressaltando que a versão 8 removeu a caixa de mensagem e deixou o texto junto com a tela esmaecida.

Mas, ao desligar, o Android ainda exibe, em PT-BR, como "Encerrando", cometendo mesmo o erro que a Microsoft fazia com o Windows até o Seven, ao invés de colocar como "Desligando".

Release 2

Conforme prometido no primeiro artigo, eis as novidades do novo lançamento oficial do projeto, disponível ao público em 13 de Junho de 2019:

A versão regular (tanto x86 como x64) foi atualizada para o kernel 4.19.50 LTS (vale ressaltar que o *bug* do Play Services persiste na versão 32 bits, contudo, desta vez, não está nem relatado como problema conhecido).

A compilação x86 teve um crescimento irrisório no consumo, em relação ao lançamento anterior, permanecendo em cerca de 1,48 GB no modo leitura-escrita.

A versão alternativa, segundo o site oficial, "recomendado para usuários

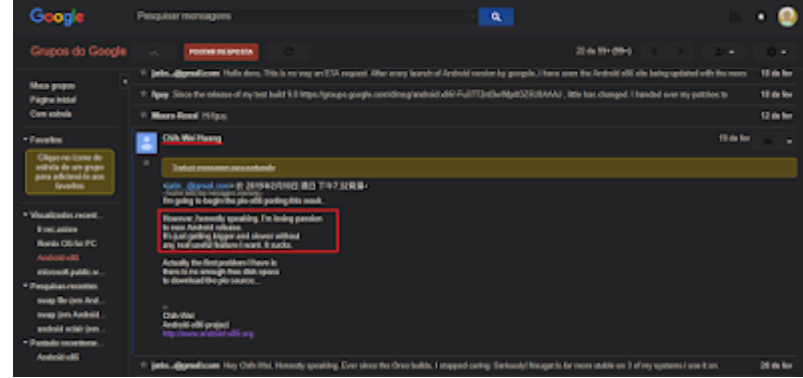
do VMWare", vem com o kernel 4.9.181 LTS (vale ressaltar que, além de ser algo inédito na história do projeto a disponibilização de uma imagem de instalação oficial alternativa, é o primeiro lançamento oficial do Oreo do projeto a vir com esta versão do Linux).

Esta, por sua vez, consome cerca de 1,98 GB de armazenamento, no modo leitura-escrita.

A principal novidade desta versão é a transferência de funcionamento da conexão Ethernet para uma conexão sem-fio virtual (mostrado acima), "a fim de melhorar a compatibilidade dos aplicativos", que nunca estiveram, de fato, preparados para funcionar com uma conexão cabeada (isto é, configurações que preveem o suporte, como a limitação do consumo de dados via Ethernet, por exemplo).

Vale ressaltar que, no passado, o projeto mantinha uma opção adicional que permitia o ajuste do dispositivo Ethernet dentro das Configurações do Android (embora, por não ser nativo, permanecia somente em inglês), entretanto, como já não existia o suporte dos aplicativos em uma conexão cabeada naquela época, o recurso se perdeu após o ICS (imagem acima) e nunca mais foi reincluído oficialmente.

Pois bem, em testes, constatei que a novidade só funciona corretamente numa instalação limpa; ao tentar fazer o *upgrade* para a nova versão, o Android não reconheceu corretamente o Wi-Fi e não foi possível se conectar à Internet.

Ao checar no modo console (acessado através da combinação de teclas ALT + F1, retornando a GUI mais tarde com ALT + F7) os dispositivos de rede, constatei que eles estavam como deveriam, exceto pela ausência do endereço IP, Broadcast e Máscara de sub-rede (por padrão em uma máquina virtual em modo NAT é, respectivamente, 10.0.2.15; 10.0.2.255; 255.255.255.0); tentei adicionar manualmente, mas não houve sucesso.

O problema foi relatado em mais de um tópico no fórum do projeto no Google Groups (aqui, aqui e aqui) e já é conhecido por Chih-Wei Huang e desenvolvedores do projeto; com base nestes tópicos, tentei alternar a conexão sem fio virtual de volta para a conexão cabeada e vice-versa, além de testar a adição da opção VIRT_WIFI = 1 ou VIRT_WIFI = 0 (detalhados no último link) no comando de inicialização do Grub e ambas as opções não funcionaram; contudo, segundo Huang, uma alternativa é excluir manualmente o arquivo */data/misc/wifi/wpa_supplicant.conf* (exige acesso leitura-escrita) para sanar o problema; no mais, a imagem acima ilustra o *upgrade* da versão R1 x86 para a R2 x64 com kernel 4.9, onde as opções Wi-Fi e Bluetooth nem são exibidos na Barra de Notificação.

Outra novidade foi a substituição do Emulador de Terminal para o aplicativo do desenvolvedor Roumen Petrov, aqui em sua versão 2.9 que, segundo o site do projeto, é o sucessor do *app* desenvolvido por Jack Palevich, que equipava as compilações do projeto até então (mas que não era atualizado há mais de 4 anos, pois, de acordo com ele, "o desenvolvimento já está completo").

No mais, o sistema vem com o patch de 5 de Junho de 2019, o driver Mesa 18.3.6, além de correções na busca de dispositivos Bluetooth, entre outros aprimoramentos. Vale ressaltar que, diferente de compilações de versões anteriores do Android, desta vez os Google Apps incluídos no sistema não foram atualizados para as versões mais recentes até a data da compilação, permanecendo nas mesmas numerações do lançamento anterior. Para todos os detalhes deste lançamento, clique aqui (em inglês).

Por último, no dia do lançamento do Oreo R2, Chih-Wei Huang respondeu a um usuário neste tópico que, após este lançamento, voltará a concentrar os esforços para o futuro lançamento de uma compilação oficial baseado no Android 9 Pie (é importante salientar que, apesar do iminente lançamento do Android Q, leva-se vários meses para, após o lançamento e adequação do código AOSP para as arquiteturas suportadas pelo projeto, fazer e revisar os *patches* e ajustes necessários, incluindo todo o *know how* das versões anteriores do Android x86 para portar no código da nova ramificação de desenvolvimento, até o lançamento de compilações oficiais, podendo levar até 1 ano ou mais para tal).

Conclusão

Após esta grande análise, pode-se dizer que, particularmente, esperava bem menos novidades nesta versão do Android (para falar a verdade, a cada versão que passa, acho que o SO da Google está ficando cada vez mais fútil em novos recursos e no limite para tentar evoluir mais), mas deu para perceber que o sistema está mais maduro, bem mais coeso e não ficando atrás das Custom ROMs. Entretanto, o excesso de branco que tomou o visual do robozinho, a fim de atrair usuários do iPhone, continua me incomodando (ainda que a possibilidade de usar temas nesta versão consiga amenizar bem), contudo, a "descoberta da roda" pela empresa de Mountain View, de que o modo escuro economiza energia, pode ser um alento para o futuro do Android.

O Android x86 como projeto, por sua vez, continua em sua evolução gradativa, tentando adaptar todo o *know how* adquirido ao longo dos 10 anos de desenvolvimento (a ser completados em 2019) a cada novo lançamento do SO da Google, mas as compilações, ainda assim, acabam persistindo com erros antigos, como o caso do Play Services instável na arquitetura 32 bits, a falta de fluidez em certos cenários, entre outros. Talvez a disponibilização periódica de novos sabores mais atrapalha do que ajuda o desenvolvimento do projeto, onde até o desenvolvedor que fundou o projeto, Chih-Wei Huang, já não possui o mesmo encanto pelo SO da Google.

O líder do Android x86 desabafou, na época da liberação do código AOSP do Oreo, dizendo: "Honestamente, eu estou perdendo o entusiasmo para o novo lançamento do Android; cada vez ficando maior e mais lento, sem qualquer recurso útil que o sistema necessita; isto é irritante", em tradução livre, o que abre um sinal de alerta para o futuro do projeto.

Pelo menos, com o projeto cada vez mais conhecido, as alternativas aumentaram, engajando um pouco mais de pessoas para o desenvolvimento do SO da Google para PCs - incluindo a Intel, com o Projeto Celadon - trazendo a expectativa de compilações mais estáveis e melhores de se utilizar. No mais, espero que esta análise tenha esclarecido a contento a situação atual do projeto e dos aplicativos suportados nele, além de ser uma referência para entender melhor esta edição do Android.

Segue, abaixo, as fontes que complementam a análise:

- https://groups.google.com/forum/#!topic/android-x86/QM7bn81djwQ
- https://sempreupdate.com.br/confira-o-android-x86-que-permite-instalar-o-android-8-1-oreo-no-pc/
- https://www.edivaldobrito.com.br/rc-do-android-8-1-oreo-para-pc/
- https://pplware.sapo.pt/smartphones-tablets/android/android-oreo-chegou-ao-pc-experimentem-o-projeto-android-x86-8-1/
- https://unixuniverse.com.br/android/x86-81-oreo
- https://www.androidpolice.com/2019/01/16/stable-version-of-android-x86-8-1-oreo-now-available/
- https://www.tudocelular.com/android/noticias/n136559/android-oreo-81-lancamento-pcs-estavel-x86.html
- https://github.com/bromite/bromite/wiki/Installing-SystemWebView
- https://www.teteututors.tech/aprenda-usar-o-android-8-oreo-em-seu-pc-sem-emulador/
- https://www.how2shout.com/tools/best-android-os-for-pc-64-bit-32-bit.html

Sintam-se à vontade para comentários e levantar dúvidas. Responderei quando for possível e desde já peço desculpas por uma eventual demora. No mais, agradeço novamente pela atenção. Fico por aqui e até a próxima.

(Última atualização: 28/6/2019)